# Benutzerhandbuch

# uEye® Software Development Kit (SDK)

**uEye® USB 2.0 Kameras**

Version 2.40

Stand: März 2007



iDS
Imaging Development Systems

**Dimbacher Straße 6**
**D-74182 Obersulm**
**Fax: +49/(0)7134/96196-99**
**eMail: sales@ids-imaging.de**

**Vorwort**

Die IDS Imaging Development Systems GmbH hat dieses Handbuch mit aller Sorgfalt erstellt. Es kann jedoch keine Garantie in Bezug auf Inhalt, Vollständigkeit und Qualität der Angaben in diesem Handbuch übernommen werden. Der Inhalt dieses Handbuches wird gepflegt und den aktuellen Gegebenheiten angepasst. Weiterhin können wir nicht gewähren, dass selbst unter Einhaltung der Spezifikationen dieses Produkt störungsfrei arbeitet.

In keinem Falle können wir eine Gewähr dafür übernehmen, dass mit dem Erwerb dieses Produktes ein bestimmtes Anwendungsziel erreicht werden kann.

Im Rahmen der gesetzlichen Möglichkeiten ist die Haftung für unmittelbare Schäden, Folgeschäden und Drittschäden, die aus dem Erwerb dieses Produktes resultieren, ausgeschlossen. Die Haftung ist in jedem Falle auf den Produktpreis beschränkt.



Stand: März 2007

**Sicherheitshinweise**
Wir weisen darauf hin, dass der Inhalt dieser Betriebsanleitung nicht Teil einer früheren oder bestehenden Vereinbarung, Zusage oder eines Rechtsverhältnisses ist oder diese abändern soll. Sämtliche Verpflichtungen der IDS Imaging Development Systems GmbH ergeben sich aus dem jeweiligen Kaufvertrag, der auch die vollständige und allein gültige Gewährleistungsregelung enthält. Diese vertraglichen Gewährleistungsbestimmungen werden durch die Ausführung dieser Betriebsanleitung weder erweitert noch beschränkt. Sollten Sie weitere Informationen zu diesem Gerät wünschen oder sollten besondere Probleme auftreten, die in der Betriebsanleitung nicht ausführlich genug behandelt werden, können Sie sich an Ihren Händler oder Errichter wenden.



**Kontaktaufnahme**

Besuchen Sie unsere Internetseite. Hier erhalten Sie die neuesten Treiber und Informationen zu unserer Soft- und Hardware sowie zu unseren Partnern und Händlern.

Internet: http://www.ueye.de
http://www.ids-imaging.de

Anschrift: IDS Imaging Development Systems GmbH
Dimbacher Strasse 6
D-74182 Obersulm

Fax: 07134/96196-99

Email: Vertrieb: sales@ids-imaging.de
Support: support@ids-imaging.de

# Inhaltsverzeichnis

**1. Einleitung ..... 1**

**2. Programmierung ..... 2**

2.1. Programmierung unter Visual C++ 6.0, 7.0 und 7.1 ..... 2

2.2. Programmierung unter Visual Basic ..... 2

2.3. CAMINFO Datenstruktur des EEPROMS ..... 2

2.4. Farb- und Speicherformate ..... 3

2.5. Bildausgabemodi ..... 4

**3. Funktionsblöcke ..... 7**

3.1. Initialisierung und Terminierung ..... 7

3.2. Bilderfassung und Speichermanagement ..... 7

3.3. Auswahl der Betriebsmodi und Rücklesen der Einstellungen ..... 8

3.4. Double- und Mehrfach-Buffering ..... 9

3.5. Lesen und Schreiben des EEPROMS ..... 9

3.6. Speichern und Laden von Bildern ..... 9

3.7. Bildausgabe ..... 9

3.8. Zusätzliche DirectDraw-Funktionen ..... 10

3.9. Event Handling (Interrupt gesteuerter Bildeinzug) ..... 10

3.10. Steuerung der Ein- / Ausgänge ..... 12

3.11. $I^2C$ Funktionen (nur uEyeLE) ..... 12

3.12. Memory Handling der Kamera ..... 13

3.13. Gültigkeit in Darstellungsmodi ..... 20

3.14. Nicht unterstütze Funktionen ..... 22

**4. Beschreibung der Funktionen ..... 23**

4.1. is_AddToSequence ..... 24

4.2. is_AllocImageMem ..... 25

4.3. is_CameraStatus ..... 26

4.4. is_CaptureVideo ..... 27

4.5. is_ClearSequence ..... 28

4.6. is_ConvertImage ..... 29

4.7. is_CopyImageMem ..... 30

4.8. is_CopyImageMemLines ..... 31

4.9. is_DisableDDOverlay ..... 31

4.10. is_DisableEvent ..... 32

4.11. is_EnableAutoExit ..... 33

4.12. is_EnableDDOverlay ..... 34

4.13. is_EnableEvent ....................35
4.14. is_EnableMessage ....................36
4.15. is_ExitCamera ....................37
4.16. is_ExitEvent ....................38
4.17. is_ForceTrigger ....................39
4.18. is_FreeImageMem ....................40
4.19. is_FreezeVideo ....................41
4.20. is_GetActiveImageMem ....................42
4.21. is_GetActSeqBuf ....................43
4.22. is_GetAutoInfo ....................44
4.23. is_GetBusSpeed ....................46
4.24. is_GetCameraInfo ....................47
4.25. is_GetCameraList ....................48
4.26. is_GetCameraType ....................49
4.27. is_GetColorDepth ....................50
4.28. is_GetDC ....................51
4.29. is_GetDDOvlSurface ....................51
4.30. is_GetDLLVersion ....................52
4.31. is_GetError ....................52
4.32. is_GetExposureRange ....................53
4.33. is_GetFramesPerSecond ....................53
4.34. is_GetFrameTimeRange ....................54
4.35. is_GetGlobalFlashDelays ....................55
4.36. is_GetImageHistogram ....................56
4.37. is_GetImageMem ....................57
4.38. is_GetImageMemPitch ....................58
4.39. is_GetLastMemorySequence ....................58
4.40. is_GetMemorySequenceWindow ....................59
4.41. is_GetNumberOfCameras ....................59
4.42. is_GetNumberOfMemoryImages ....................60
4.43. is_GetOsVersion ....................60
4.44. is_GetPixelClockRange ....................61
4.45. is_GetRevisionInfo ....................62
4.46. is_GetSensorInfo ....................63
4.47. is_GetUsedBandwidth ....................64
4.48. is_GetVsyncCount ....................65
4.49. is_GetWhiteBalanceMultipliers ....................65
4.50. is_HasVideoStarted ....................66
4.51. is_HideDDOverlay ....................66

4.52. is_InitCamera ....67
4.53. is_InitEvent ....68
4.54. is_InquireImageMem ....69
4.55. is_IsMemoryBoardConnected ....69
4.56. is_IsVideoFinish ....70
4.57. is_LoadBadPixelCorrectionTable ....71
4.58. is_LoadImage ....71
4.59. is_LoadImageMem ....72
4.60. is_LoadParameters ....73
4.61. is_LockDDMem ....75
4.62. is_LockDDOverlayMem ....76
4.63. is_LockSeqBuf ....77
4.64. is_MemoryFreezeVideo ....78
4.65. is_PrepareStealVideo ....79
4.66. is_ReadEEPROM ....80
4.67. is_ReadI2C (nur uEyeLE) ....81
4.68. is_ReleaseDC ....81
4.69. is_RenderBitmap ....82
4.70. is_ResetMemory ....83
4.71. is_ResetToDefault ....83
4.72. is_SaveBadPixelCorrectionTable ....84
4.73. is_SaveImage ....85
4.74. is_SaveImageEX ....86
4.75. is_SaveImageMem ....87
4.76. is_SaveImageMemEx ....88
4.77. is_SaveParameters ....89
4.78. is_SetAllocatedImageMem ....90
4.79. is_SetAOI ....91
4.80. is_SetAutoParameter ....93
4.81. is_SetBadPixelCorrection ....96
4.82. is_SetBadPixelCorrectionTable ....97
4.83. is_SetBayerConversion ....98
4.84. is_SetBinning ....99
4.85. is_SetBlCompensation ....100
4.86. is_SetBrightness ....101
4.87. is_SetCameraID ....101
4.88. is_SetColorCorrection ....102
4.89. is_SetColorMode ....103
4.90. is_SetContrast ....104

4.91. is_SetConvertParam ....................................................................................................105
4.92. is_SetDDUpdateTime....................................................................................................106
4.93. is_SetDisplayMode........................................................................................................107
4.94. is_SetDisplayPos ..........................................................................................................108
4.95. is_SetEdgeEnhancement...............................................................................................109
4.96. is_SetErrorReport..........................................................................................................110
4.97. is_SetExposureTime ......................................................................................................111
4.98. is_SetExternalTrigger.....................................................................................................112
4.99. is_SetFlashDelay ...........................................................................................................113
4.100. is_SetFlashStrobe ........................................................................................................114
4.101. is_SetFrameRate .........................................................................................................116
4.102. is_SetGainBoost...........................................................................................................117
4.103. is_SetGamma...............................................................................................................118
4.104. is_SetGlobalShutter .....................................................................................................119
4.105. is_SetHardwareGain .....................................................................................................120
4.106. is_SetHardwareGamma ................................................................................................121
4.107. is_SetHWGainFactor.....................................................................................................122
4.108. is_SetHwnd ..................................................................................................................124
4.109. is_SetImageAOI ............................................................................................................124
4.110. is_SetImageMem ..........................................................................................................125
4.111. is_SetImagePos ............................................................................................................126
4.112. is_SetImageSize ...........................................................................................................128
4.113. is_SetIO (nur UI-1543-M)..............................................................................................129
4.114. is_SetKeyColor.............................................................................................................129
4.115. is_SetLED ....................................................................................................................130
4.116. is_SetMemoryMode ......................................................................................................131
4.117. is_SetPixelClock...........................................................................................................132
4.118. is_SetRopEffect............................................................................................................133
4.119. is_SetSaturation ...........................................................................................................134
4.120. is_SetSubSampling.......................................................................................................135
4.121. is_SetTestImage ..........................................................................................................136
4.122. is_SetTriggerDelay.......................................................................................................137
4.123. is_SetWhiteBalance .....................................................................................................138
4.124. is_SetWhiteBalanceMultipliers......................................................................................139
4.125. is_ShowDDOverlay .......................................................................................................139
4.126. is_StealVideo ...............................................................................................................140
4.127. is_StopLiveVideo..........................................................................................................141
4.128. is_TransferImage .........................................................................................................142
4.129. is_TransferMemorySequence .......................................................................................143

 

4.130. is_UnlockDDMem ........ 144
4.131. is_UnlockDDOverlayMem ........ 144
4.132. is_UnlockSeqBuf ........ 145
4.133. is_UpdateDisplay ........ 145
4.134. is_WriteEEPROM ........ 146
4.135. is_WriteI2C (nur uEyeLE) ........ 146
**5. Fehlermeldungen ........ 147**
**Service und Support ........ 149**
**Abbildungsverzeichnis ........ 151**
**Tabellenverzeichnis ........ 151**

# 1. Einleitung

Vielen Dank für den Kauf einer uEye USB2.0 Kamera der *Fa. IDS Imaging Development Systems GmbH*.

Die uEye Serie besteht aus Monochrom- und Farb- Kameras für industrielle, medizinische und multimediale Anwendungen. Unterstützt werden USB Schnittstellen nach dem Standard 2.0 (USB1.1 nur mit Memoryboard). Die Bilder sind abhängig vom spezifischen Kameramodell in 8-Bit monochromer- bzw. 24-Bit Echtfarbqualität verfügbar.
Zur Einbindung der uEye Kamera in eigene Programme unter Windows 2000, Windows XP und Linux ist ein SDK (Software Development Kit) im Lieferumfang enthalten.

Dieses Handbuch beschreibt die Funktionen des uEye Software Development Kit (SDK).

> Das Software Development Kit von uEye ist, bis auf zusätzliche Funktionalitäten bzw. bauart- und anschlussbedingte Änderungen, nahezu identisch mit dem SDK der FALCON bzw. EAGLE Framegrabber.

Bitte lesen Sie auch die Datei LIESMICH.TXT, die sich auf der Installations-CD befindet. Hier sind zusätzliche Informationen zu finden, die in dieser Ausgabe des Handbuches eventuell noch nicht enthalten sind.

Wir wünschen Ihnen viel Erfolg mit diesem Produkt.

# 2. Programmierung

## 2.1. Programmierung unter Visual C++ 6.0, 7.0, 7.1 und 8.0

Bitte beachten Sie, dass Microsoft das Format der .lib-Datei ab Version 6.0 geändert hat. Die Funktionen zur uEye Kamerafamilie wurden mit Visual C++ 7.1 erstellt. Somit ist das lib-File *uEye_api.lib* nur mit einem Compiler der Version 6.0 oder höher zu verwenden.

## 2.2. Programmierung unter Visual Basic

Die Funktionen des Software Development Kits sind mit der Aufrufkonvention „_cdecl" exportiert. Visual Basic benötigt jedoch Funktionen mit der Konvention „_stdcall" (Pascal-Konvention). Sie können die uEye-Funktionen direkt aus Visual Basic aufrufen, wenn Sie den Präfix is_<Funktionsname> durch den Präfix iss_<Funktionsname> ersetzen. Alle in diesem Handbuch beschriebenen Funktionen is_<Funktionsname> sind _cdecl-Funktionen. Zu allen diesen Funktionen existieren parallel _stdcall-Funktionen iss_<Funktionsname>. Übergabeparameter sowie Rückgabewerte sind ebenfalls identisch.

## 2.3. CAMINFO Datenstruktur des EEPROMS

Mit Hilfe der Funktion *is_GetCameraInfo()* können die in der Kamera fest hinterlegten Daten ausgelesen werden. Die Datenstruktur ist 64 Byte lang und wie folgt aufgebaut:

| | | |
|---|---|---|
| Char | SerNo[12] | Seriennummer der Kamera |
| Char | ID[20] | z.B.: „IDS GmbH" |
| Char | Version[10] | z.B.: „V1.00" oder nachfolgende Versionen |
| Char | Date[12] | „01.08.2004" Systemdatum des Endtests |
| unsigned char | Select | Kamera-ID |
| unsigned char | Type | Kamera-Typ<br>64 = uEye USB2.0 |
| Char | Reserved[8] | reserviert |

*Tabelle 1: CAMINFO Datenstruktur des EEPROMS*

## 2.4. Farb- und Speicherformate

Die von der uEye Kamera unterstützten Farbformate bewirken unterschiedliche Speicherformate. Aus der nachfolgenden Tabelle entnehmen Sie die Reihenfolge der Anordnung im Speicher:

| | Pixel Data | | | |
|---|---|---|---|---|
| Format | Byte 3 [Bit 31:24] | Byte 2 [Bit 23:16] | Byte 1 [Bit 15:8] | Byte 0 [Bit 7:0] |
| RGB32 | | | | |
| RGB24 | | | | |
| RGB16 | | | | |
| RGB15 | | | | |
| UYVY | Y1 | V0 | Y0 | U0 |
| Y8 | Y3 | Y2 | Y1 | Y0 |

*Tabelle 2: Farb- und Speicherformate*

*Abb. 1:Prinzipieller Aufbau des Bayer-Pattern*

Bei den Datenformaten RGB16 und RGB15 werden von den internen 8-Bit R, G- und B-Farben die jeweils oberen Bits verwendet.

## 2.5. Bildausgabemodi

**Bitmap Modus (BMP/DIB)**
Nach dem Starten des Demoprogramms *uEye Demo,* welches im nächsten Kapitel beschrieben wird, zunächst der Bitmap-Modus aktiv. Hier wird das über die uEye USB2.0 Kamera eingelesene Bild in den Hauptspeicher des PCs geschrieben. Die Bildausgabe muss der Anwender selbst handhaben in dem unter Programmkontrolle und unter Belastung der CPU das Bild in ein Bitmap gewandelt und in die VGA-Karte kopiert wird. Der große Vorteil dieses Modus ist die Kompatibilität zu allen VGA-Karten und der Möglichkeit des Zugriffs auf die Bilddaten im Speicher. Overlay-Funktionen müssen vom Anwender programmiert werden. Da Windows die Steuerung der Bildausgabe übernimmt, kann das Bild von beliebig anderen Fenstern und Dialogboxen ganz oder teilweise überlagert werden.

**Bitmap Modus**

*Abb. 2: Bitmap Modus*

**DirectDraw BackBuffer Modus (unter LINUX nicht verfügbar)**
In diesem Modus werden die Bilddaten in den nicht sichtbaren Bereich der VGA-Karte geschrieben. Voraussetzungen sind hierbei: Installierter DirectDraw-Treiber, ausreichend Speicher auf der VGA-Karte und BackBuffer-Support des VGA-Karten Herstellers. Im BackBuffer Modus mit Overlay werden 3 nichtsichtbare Bildbuffer verwendet:

- BackBuffer
- OverlayBuffer
- MixBuffer

Die drei Buffer besitzen jeweils die Größe: Video_X * Video_Y * Farbtiefe (in Bytes pro Pixel). Das Videobild wird in den BackBuffer geschrieben. Das Overlay kann in den OverlayBuffer gezeichnet werden (siehe auch *4.28 is_GetDC* und *4.68 is_ReleaseDC*). Das Overlay wird nicht direkt eingeblendet. Es muss zuvor mit *is_ShowDDOverlay()* (siehe *4.125 is_ShowDDOverlay*) sichtbar gemacht werden. Es benutzt als Color-Key die Farbe schwarz, so dass eine Overlay-Grafik keine schwarze Farbe enthalten kann (Behelf über eine nahezu schwarze Farbe, z.B. Dunkelblau).
BackBuffer und OverlayBuffer werden zusammen in den MixBuffer geschrieben. Dabei werden die Overlaydaten dem Videobild überlagert. Anschließend wird der MixBuffer in den sichtbaren Bereich der VGA-Karte kopiert. Das Ergebnis ist eine Live-Bilddarstellung mit überlagertem Text- und Grafik-Overlay.  Die Bildwiederholrate und die Belastung der CPU hängen von der

 

eingestellten Farbtiefe und vom Ort (Systemspeicher des PCs oder Bildspeicher der VGA-Karte) des BackBuffers ab.
Der Treiber versucht die Buffer direkt in der VGA-Karte zu allokieren, um den Hochgeschwindigkeits-Bildtransfer der VGA-Karte beim Mischen der drei Buffer auszunutzen. Können die Buffer nicht in der VGA-Karte allokiert werden, muss auf den Systemspeicher ausgewichen werden. Der Bildtransfer aus dem Systemspeicher ist jedoch langsamer oder unter Umständen auch gar nicht möglich (je nach Grafikkarte). Eine Skalierung des Videobildes ist im BackBuffer-Modus nicht möglich.
Der BackBuffer Modus wird wie folgt gesetzt:
Mode = *IS_SET_DM_DIRECTDRAW* | *IS_SET_DM_BACKBUFFER*

*Abb. 3: DirectDraw BackBuffer Modus*

**DirectDraw Overlay-Surface-Modus (Unter LINUX nicht verfügbar)**
In diesem Modus wird ein Live-Bild bei gleichzeitiger Darstellung von Overlaydaten erreicht. Das Videobild wird in einen nicht sichtbaren Bereich der VGA-Karte digitalisiert. Dieser Bereich muss sich immer auf der VGA Karte befinden. Durch Definition einer Keying-Farbe und Zeichnen dieser Farbe in das Bildausgabefenster wird überall dort, wo sich diese Keying-Farbe im Ausgabefenster befindet, das Videobild eingeblendet. Wird der Fensterbereich mit der Keying-Farbe gefüllt, so erscheint das Videobild. Entsprechend bleiben Grafik/Textdaten erhalten, die nicht mit der Keying-Farbe gezeichnet wurden. Dadurch entsteht ein non-destructive Overlay. Das Einblenden erfolgt durch den VGA-Chip und benötigt nahezu keine Rechenzeit. Dieser Modus wird nicht von allen VGA-Chips unterstützt und ist häufig nur im YUV Modus möglich. Bestes Text/Grafik-Overlay mit Fensterüberlagerung erreicht man durch Setzen des nachfolgenden Videomodus:
Mode = *IS_SET_DM_DIRECTDRAW* | *IS_SET_DM_ALLOW_OVERLAY*

Wenn das Videobild auf die Fenstergröße skaliert werden soll (auch Vergrößerung möglich), dann kann dies wie folgt erreicht werden:
Mode = *IS_SET_DM_DIRECTDRAW* | *IS_SET_DM_ALLOW_OVERLAY* |
*IS_SET_DM_ALLOW_SCALING*

*Abb. 4: DirectDraw Overlay-Surface-Modus*

Beim FALCON Framegrabber wurde der BackBuffer-Modus durch das Setzen des Parameters IS_SET_DM_DIRECTDRAW aktiviert. Der BackBuffer-Modus der uEye-Kameras wird wie folgt aktiviert: IS_SET_DM_DIRECTDRAW | IS_SET_DM_BACKBUFFER.

# 3. Funktionsblöcke

## 3.1. Initialisierung und Terminierung

| Funktionsliste | |
|---|---|
| is_ExitCamera | Schließen der Kamera und Freigeben der mit dem SDK angelegten Bildspeicher |
| is_InitCamera | Initialisierung der Hardware |
| is_LoadParameters | Laden und anwenden der Kamera Parameter |
| is_SaveParameters | Speichert die aktuellen Kamera Parameter |
| is_SetCameraID | Setzt eine neue Kamera ID |

*Tabelle 3: Funktionsliste Initialisierung und Terminierung*

## 3.2. Bilderfassung und Speichermanagement

| Funktionsliste | |
|---|---|
| is_AllocImageMem | Bildspeicher anlegen |
| is_CaptureVideo | Live Video erfassen |
| is_ConvertImage | Wandelt ein RAW Bayer Bild in das gewünschte Format um |
| is_CopyImageMem | Bild in vom Anwender definierten Speicher kopieren |
| is_CopyImageMemLines | Ausgewählte Zeilen des Bildes in vom Anwender definierten Speicher kopieren |
| is_FreeImageMem | Einen allokierten Bildspeicher wieder freigeben |
| is_FreezeVideo | Ein Bild erfassen und auf Zielbildspeicher schreiben, Snap |
| is_GetActiveImageMem | Gibt Nummer und Adresse des aktiven Bildspeichers zurück |
| is_GetBusSpeed | Prüfen ob Kamera an einem USB 2.0 Hostcontroller angeschlossen ist |
| is_GetImageHistogram | Berechnet das Histogramm der übergebenen Abbildung |
| is_GetImageMem | Gibt den Anfangszeiger auf den Bildspeicher zurück |
| is_GetImageMemPitch | Gibt den Zeilenoffset (n) zu (n+1) zurück |
| is_HasVideoStarted | Ist Bildaufnahme gestartet? |
| is_InquireImageMem | Gibt Eigenschaften eines Bildspeichers zurück |
| is_IsVideoFinish | Ist Bildaufnahme beendet? |
| is_SaveImageMem | Bildspeicher als Bitmap speichern |
| is_SetAllocatedImageMem | Anwender stellt Speicherbereich zur Bilderfassung bereit |
| is_SetBayerConversion | Wählt Bayer Algorithmus aus |
| is_SetImageMem | Einen Bildspeicher als aktiv setzen |
| is_SetTestImage | Aktiviert Testbilder |
| is_StopLiveVideo | Beednet die Aufnahme (kontinuierlich oder Einzelbild) |

*Tabelle 4: Funktionsliste Bilderfassung und Speichermanagement*

## 3.3. Auswahl der Betriebsmodi und Rücklesen der Einstellungen

| Funktionsliste | |
|---|---|
| is_CameraStatus | Liefert Event-Zähler und Counter-Werte |
| is_GetAutoInfo | Liefert Statusinformationen der Autofunktionalität |
| is_GetCameraList | Liefert Informationen über die angeschlossenen Kameras |
| is_GetCameraType | Liefert den Kamera-Typ |
| is_GetColorDepth | Aktuellen Farbmodus der VGA-Karte ermitteln |
| is_GetDLLVersion | Gibt die Version der ueye_api.dll zurück |
| is_GetError | Fehlermeldung abfragen |
| is_GetExposureRange | Belichtungsbereich ermitteln |
| is_GetFramesPerSecond | Gibt aktuelle Framerate im Livemodus zurück |
| is_GetFrameTimeRange | Frameraten-Bereich ermitteln |
| is_GetNumberOfCameras | Ermittelt die Anzahl im System vorhandener Kameras |
| is_GetOsVersion | Betriebssystem erfragen |
| is_GetPixelClockRange | Gibt einstellbaren Bereich für den Pixeltakt zurück |
| is_GetUsedBandwidth | Summe der aktuell eingestellten Pixeltakte |
| is_GetVsyncCount | Auslesen des VSYNC-Zählers |
| is_GetWhiteBalanceMultipliers | Auslesen der aktuellen Parameter des Weißabgleichs |
| is_LoadBadPixelCorrectionTable | Eine benutzerdefinierte Hotpixel Liste aus einer Datei laden |
| is_PrepareStealVideo | Setzt den Stehlen Modus |
| is_ResetToDefault | Rücksetzen der Kameraparameter auf Standardwerte |
| is_SaveBadPixelCorrectionTable | Speichert die aktuelle, benutzerdefinierte HotpixelListe |
| is_SetAOI | Größe und Position eines AOI setzen |
| is_SetAutoParameter | Aktiviert/deaktiviert Gain/Shutter/Whitebalance Autofunktionen |
| is_SetBadPixelCorrection | Ein-/ausschalten und parametrieren der Hotpixelkorrektur |
| is_SetBadPixelCorrectionTable | Übergibt dem SDK eine benutzerdefinierte Hotpixel Liste |
| is_SetBinning | Einstellen der Binning Modi |
| is_SetBlCompensation | Ein-/Ausschalten der Blacklevel Kompensation |
| is_SetBrightness | Einstellen der Bildhelligkeit (digitale Nachbearbeitung) |
| is_SetColorCorrection | Farbkorrektur einstellen |
| is_SetColorMode | Farbmodus auswählen |
| is_SetContrast | Einstellen des Kontrastes (digitale Nachbearbeitung) |
| is_SetConvertParam | Konvertierungsparameter für ein RAW Bayer Bild |
| is_SetDisplayMode | Auswahl des Modus der Bilddarstellung |
| is_SetEdgeEnhancement | Kantenfilter einstellen |
| is_SetErrorReport | Fehlerausgabe aktivieren/deaktivieren |
| is_SetExposureTime | Einstellen der Belichtungszeit |
| is_SetFrameRate | Einstellen der Framerate |
| is_SetGainBoost | Aktiviert/deaktiviert zusätzliche Hardwareverstärkung |
| is_SetGamma | Einstellen des Gammawertes (digitale Nachbearbeitung) |
| is_SetGlobalShutter | Aktiviert/deaktiviert den *Global Start Shutter*. |
| is_SetHardwareGain | Hardware Verstärkung einstellen. |
| is_SetHardwareGamma | Aktiviert/deaktiviert die Gammaregelung der Kamera. |
| is_SetHWGainFactor | Steuerung der Verstärker einer Kamera |
| is_SetHwnd | Fensterhandle für die Bildausgabe unter DirectDraw |
| is_SetImageAOI | Bildposition und Größe festlegen |
| is_SetImagePos | Bildposition innerhalb des Bildfensters bestimmen |
| is_SetImageSize | Bildgröße festlegen |
| is_SetLED | LED ein-/ausschalten |
| is_SetPixelClock | Pixeltakt einstellen |

| | |
|---|---|
| is_SetRopEffect | Echtzeit-Bildmanipulationseffekte einstellen |
| is_SetSaturation | Einstellen der Software-Bildsättigung |
| is_SetSubSampling | Einstellen der Subsampling Modi |
| is_SetWhiteBalance | Weißabgleich aktivieren |
| is_SetWhiteBalanceMultipliers | Einstellen der Weißabgleichparameter |

*Tabelle 5: Funktionsliste Auswahl der Betriebsmodi und Rücklesen der Einstellungen*

## 3.4. Double- und Mehrfach-Buffering

| Funktionsliste | |
|---|---|
| is_AddToSequence | Bildspeicher in Sequenz-Liste aufnehmen |
| is_ClearSequence | Komplette Sequenz-Liste löschen |
| is_GetActSeqBuf | Aktuell verwendeten Bildspeicher der Sequenz ermitteln |
| is_LockSeqBuf | Bildspeicher der Sequenz vor Überschreiben schützen |
| is_UnlockSeqBuf | Bildspeicher der Sequenz zum Überschreiben freigeben |

*Tabelle 6: Funktionsliste Double- und Mehrfach-Buffering*

## 3.5. Lesen und Schreiben des EEPROMS

| Funktionsliste | |
|---|---|
| is_GetCameraInfo | Holt sich die ab Werk eingebrannten Informationen |
| is_GetRevisionInfo | Revisionsinformationen der einzelnen uEye Komponenten |
| is_GetSensorInfo | Auslesen der Sensorinformationen |
| is_ReadEEPROM | Liest eigene Daten aus EEPROM |
| is_WriteEEPROM | Beschreibt das EEPROM mit eigenen Daten |

*Tabelle 7: Funktionsliste Lesen und Schreiben des EEPROMS*

## 3.6. Speichern und Laden von Bildern

| Funktionsliste | |
|---|---|
| is_LoadImage | Bitmap Datei in den aktuellen Bildspeicher laden |
| is_LoadImageMem | Bild aus Datei in den aktuellen Bildspeicher laden |
| is_SaveImage | Speichert Videobild als BMP-Datei ab |
| is_SaveImageEX | Speichert Videobild in einer Datei ab |
| is_SaveImageMem | Speichert Bildspeicher als BMP-Datei ab |
| is_SaveImageMemEx | Speichert Bildspeicher in einer Datei ab ab |

*Tabelle 8: Funktionsliste Speichern und Laden von Bildern*

## 3.7. Bildausgabe

| Funktionsliste | |
|---|---|
| is_RenderBitmap | Gibt ein Bild eines Bildspeichers in einem Fenster aus |
| is_SetDisplayPos | Ermöglicht die Verschiebung der Bildausgabe |
| is_UpdateDisplay | Bildschirm Refresh bei DirectDraw |

*Tabelle 9: Funktionsliste Bildausgabe*

## 3.8. Zusätzliche DirectDraw-Funktionen

| Funktionsliste | |
|---|---|
| is_DisableDDOverlay | Deaktiviert den Overlay-Modus |
| is_EnableDDOverlay | Aktiviert den Overlay-Modus |
| is_GetDC | Holt Device Context Handle des Overlayspeichers |
| is_GetDDOvlSurface | Gibt Zeiger auf DirectDraw Surface zurück |
| is_HideDDOverlay | Blendet das Overlay aus |
| is_LockDDMem | Freigabe des Zugriffs auf Back Buffer der VGA-Karte |
| is_LockDDOverlayMem | Gibt den Zugriff auf den Overlayspeicher frei |
| is_ReleaseDC | Freigabe Device Context Handle des Overlays |
| is_SetDDUpdateTime | Timer-Intervall für den Update-Zyklus setzen |
| is_SetKeyColor | Setzt die Keying-Farbe für die Overlay-Darstellung |
| is_ShowDDOverlay | Overlay anzeigen |
| is_StealVideo | Stiehlt ein Bild aus einem DirectDraw Live Modus und legt dieses im Bildspeicher im RAM ab |
| is_UnlockDDMem | Sperrt Zugriff auf den Back Buffer der VGA-Karte |
| is_UnlockDDOverlayMem | Sperrt den Zugriff auf den Overlayspeicher |

*Tabelle 10: Funktionsliste Zusätzliche DirectDraw-Funktionen*

## 3.9. Event Handling (Interrupt gesteuerter Bildeinzug)

| Funktionsliste | |
|---|---|
| is_DisableEvent | Sperren der Event Objekte |
| is_EnableEvent | Freigabe der Event Objekte |
| is_EnableMessage | Ein-/ausschalten der Windows Benachrichtigungen |
| is_ExitEvent | Verlassen des Event Handlers |
| is_InitEvent | Einrichten des Event Handlers |
| is_EnableAutoExit | Kamera Ressourcen werden beim Abziehen des USB-Kabels automatisch freigegeben |

*Tabelle 11: Funktionsliste Event Handling*

### Events bei Einzeltriggeraufnahme

is_SetExternalTrigger(,IS_SET_TRIG_HI_LO)
is_FreezeImage()

Trigger
Exposure
Transfer
Post Processing
Delay
IS_TRIGGER
IS_FRAME

*Abb. 5: Events bei Einzeltriggeraufnahme*

### Events im Livemodus (Sequenz mit 3 Bildern)

is_SetDisplayMode(,IS_SET_DM_DIRECTDRAW)
is_CaptureVideo()
is_StealVideo()

Exposure
Transfer
Post Processing
IS_FRAME
IS_FRAME
IS_FRAME
IS_FRAME
IS_SEQUENCE
IS_STEAL_VIDEO

*Abb. 6: Events im Livemodus*

**Events im Memorymodus**



*Abb. 7: Events im Memorymodus*

## 3.10. Steuerung der Ein- / Ausgänge

| Funktionsliste | |
|---|---|
| is_ForceTrigger | Auslösen eines Hardwaretriggers. |
| is_GetGlobalFlashDelays | Verzögerungs- und Einschaltzeit des Blitzausgangs bei Rolling-Shutter Sensoren ermitteln. |
| is_SetExternalTrigger | Aktiviert den externen Triggereingang oder liest den anliegenden Signalpegel aus. |
| is_SetFlashDelay | Verzögerungs- und Einschaltzeit des Blitzausgangs setzen. |
| is_SetFlashStrobe | Setzt den Flash-Strobe-Ausgang (Blitzansteuerung) oder die statische Ausgabe |
| is_SetIO (nur UI-1543-M) | Setzen der zusätzlichen digitalen Ausgänge |
| is_SetTriggerDelay | Verzögerungszeit des Triggersignals einstellen |

*Tabelle 12: Funktionsliste Steuerung der Ein- / Ausgänge*

## 3.11. I²C Funktionen (nur uEyeLE)

| Funktionsliste | |
|---|---|
| is_ReadI2C (nur uEyeLE) | Daten über den I²C-Bus lesen |
| is_WriteI2C (nur uEyeLE) | Daten über den I²C-Bus schreiben |

*Tabelle 13: Funktionsliste I²C-Funktionen*

## 3.12. Memory Handling der Kamera

**Funktionsliste:**

| | |
|---|---|
| is_GetLastMemorySequence | Liefert ID der zuletzt aufgenommenen Sequenz im Memoryboard |
| is_GetMemorySequenceWindow | Liefert Fenstergröße zu einer angegebenen Memoryboard Sequenz |
| is_GetNumberOfMemoryImages | Liefert Anzahl an gültigen Bilder, die sich innerhalb der angegebenen Sequenz-ID im Kameraspeicher befinden |
| is_IsMemoryBoardConnected | Prüfen ob das optionale Memoryboard vorhanden ist |
| is_MemoryFreezeVideo | Einzelbild über das Memoryboard aufnehmen |
| is_ResetMemory | Speicher des Memoryboards löschen |
| is_SetMemoryMode | Aktiviert das optionale Memoryboard |
| is_TransferImage | 1 Bild aus dem Kameraspeicher einlesen |
| is_TransferMemorySequence | Mehrere Bilder aus dem Kameraspeicher in eine SDK Sequenz einlesen |

*Tabelle 14: Funktionsliste Memory Handling*

In Verbindung mit dem Memory-Erweiterungsmodul für die uEye Kamerafamilie besteht die Möglichkeit neuer Aufnahmevarianten. Zur Steuerung werden neue SDK Funktionen zur Verfügung gestellt, bzw. werden die Funktionalitäten bestehender Funktionen erweitert.

In Abhängigkeit ob ein Triggersignal die Aufnahme beendet oder startet ergeben sich für die zwei Aufzeichnungsmodi die Bezeichnungen

- Pre-Trigger und
- Post-Trigger Modus.

**Pre-Trigger**

In diesem Modus zeichnet das Memoryboard kontinuierlich Bilder auf. Beim Auslösen des Triggers wird die Aufnahme beendet und es stehen die n letzten Bilder im Kameraspeicher zur Verfügung.

- Vorbereitung
  Mit der Funktion *is_SetExternalTrigger* wird der Triggereingang der Kamera aktiviert. Die Kamera wird mit der Funktion *is_SetMemoryMode* für die Speicherung vorbereitet.

> Die Reihenfolge der Funktionen *is_SetMemoryMode* und *is_SetExternalTrigger* ist beliebig.
>
> Wird mit *is_SetExternalTrigger*(hCamera, IS_SET_TRIG_OFF) der Trigger deaktiviert, so wird die Aufnahme direkt beim Aufruf von *is_StopLiveVideo* beendet.

Als Parameter bekommt diese Funktion eine Zahl mit, die die Anzahl an Bildern in einem Ringspeicher beschreibt, welcher bis zum Eintreten des Triggers zyklisch überschrieben wird.
Die maximal mögliche Anzahl Bilder, die im Speicher gehalten werden können ist abhängig von der eingestellten Bildgröße, weshalb diese nach Aktivierung des Memorymodus nicht mehr zu ändern ist.

- Aufnahme
  Mit dem Aufruf von *is_CaptureVideo* startet die Aufnahme. Es werden n Bilder in den Speicher geschrieben  und beim Eintreffen weiterer Bilder wird das jeweils älteste überschrieben.

  Nach dem Aufruf von *is_StopLiveVideo* mit einem optionalen timeout Wert werden so lange weiter Bilder aufgenommen und in den Kameraspeicher geschrieben, bis ein Triggersignal registriert wird. Das derzeit aufgenommene Bild wird noch fertig geschrieben. Danach sind die n  zuletzt aufgenommenen Bilder im Speicher abrufbar. Die aufgenommene Sequenz hat eine eindeutige Sequenz Id erhalten, unter der die Bilder indiziert werden können. Diese Sequenz Id ist mit dem Befehl *is_GetLastMemorySequence* nach Beenden der Aufnahme abrufbar.
  Läuft die mit timeout spezifizierte Zeit ab, bevor das Triggersignal ausgelöst wurde, wird ein Fehler zurückgegeben. Eventuell in dieser Sequenz bereits aufgenommene Bilder werden verworfen und die Sequenzdaten sind ungültig *is_GetLastSequence()* gibt in diesem Fall 0 zurück (0 ist keine gültige Sequenz Id).
- Codebeispiel:

```
int nNumberOfImages = 5, nSequence = 0;

is_SetExternalTrigger(hCamera, IS_TRIG_HI_LO);

//wenn die Speicherung so vieler Bilder möglich ist,
if (is_SetMemoryMode(hCamera, nNumberOfImages, 0) == IS_SUCCESS)
{
       // starte Bildaufnahme
       is_CaptureVideo(hCamera, IS_WAIT);

       // warten auf Triggersignal
       is_StopLiveVideo(hCamera, IS_WAIT);

       //Sequenz ist gültig?
       is_GetLastMemorySequence(hCamera, &nSequence);
       if (nSequence != 0)
              is_TransferImage(hCamera, 0, nSequence, 3, 0);
       ...
}
```

Start
SetMemoryMode (N)
SetExternalTrigger
SetExternalTrigger
SetMemoryMode (N)
CaptureVideo
Save Image to Memory
- assign SequenceID
- assign ImageIndex n
- increment ImageIndex n
- save Image at position n%N
...
Image [n-1%N]
Image [n-2%N]
Image [n%N]
...
StopLiveVideo (Timeout)
no
yes
Timeout ?
no
yes
Trigger ?
no
yes
NO_SUCCESS
SUCCESS

*Abb. 8: Pre-Trigger Modus*

**Post-Trigger**

In diesem Modus wird mit der Bildaufnahme erst begonnen, nachdem das Triggersignal registriert wurde.

- Vorbereitung
  Mit der Funktion *is_SetExternalTrigger* wird der Triggereingang der Kamera aktiviert.
  Die Kamera wird mit der Funktion *is_SetMemoryMode* für die Speicherung vorbereitet. Neben der Anzahl an Bildern die aufgenommen werden soll, wird hier noch die Zeit zwischen zwei Aufnahmen angegeben. Die maximal mögliche Anzahl Bilder, die im Speicher gehalten werden können ist abhängig von der eingestellten Bildgröße, weshalb diese nach Aktivierung des Memorymodus nicht mehr zu ändern ist.

Die Reihenfolge der Funktionen *is_SetMemoryMode* und *is_SetExternalTrigger* ist beliebig. Wird mit *is_SetExternalTrigger*(hCamera, IS_SET_TRIG_OFF) der Trigger deaktiviert, so wird die Aufnahme direkt beim Aufruf von *is_FreezeVideo* gestartet.

- Aufnahme
  Dieser Modus wird durch den Aufruf von *is_FreezeVideo* aktiviert. Ein optionaler timeout Wert gibt an, wie lange auf das Triggersignal gewartet werden soll. So lange dieses Signal nicht eintrifft, befindet sich die Kamera in Bereitschaft. Beim Auftreten eines Triggersignals startet die Bildaufnahme und es wird die mit *is_SetMemoryMode* angegebene Anzahl an Bildern aufgenommen. Ist diese Anzahl Bilder fertig aufgenommen, wird eine eindeutige Sequenz Id vergeben, unter der diese Bilder zu einem späteren Zeitpunkt indiziert werden können. Läuft jedoch der timeout ab, bevor alle Bilder aufgenommen wurden, ist die gesamte Sequenz ungültig (Sequenz Id = 0) und ein Fehler wird gemeldet. Die Id einer Sequenz kann nachdem alle Bilder aufgenommen wurden mit dem Befehl *is_GetLastMemorySequence* abgerufen werden.
- Codebeispiel

```
int nNumberOfImages = 5, nSequence = 0;

is_SetExternalTrigger(hCamera, IS_TRIG_HI_LO);

//wenn die Speicherung so vieler Bilder möglich ist,
if (is_SetMemoryMode(hCamera, nNumberOfImages, 100) == IS_SUCCESS)
{
        // starte Bildaufnahme und warten auf Triggersignal
        is_FreezeVideo(hCamera, IS_WAIT);

        //Sequenz ist gültig?
        is_GetLastMemorySequence(hCamera, &nSequence);
        if (nSequence != 0)
                is_TransferImage(hCamera, 0, nSequence, 1, 0);
}
```

Start
SetMemoryMode (N, t)
SetExternalTrigger
SetExternalTrigger
SetMemoryMode (N, t)
FreezeVideo (timeout)
Ready
Save Image to Memory
- assign SequenceID
- assign ImageIndex n
- increment ImageIndex n
- save Image at position n%N
...
Image [n-1%N]
Image [n-2%N]
Image [n%N]
...
no
yes
Trigger ?
no
Timeout ?
yes
no
Timeout ?
Wait dt
no
n < N ?
yes
Timeout ?
no
SUCCESS
yes
yes
NO_SUCCESS

*Abb. 9: Post-Trigger Modus*

**Mehrere Sequenzen**
Es ist durchaus möglich mehrere Sequenzen aufzunehmen, ohne die aufgenommenen Bilder zwischen zwei Sequenzen aus dem Kameraspeicher auslesen zu müssen.
Für ältere Sequenzen kann allerdings nicht garantiert werden, dass diese noch im Kameraspeicher präsent sind.
Eine neue Sequenz speichert ihre Bilder vornehmlich in freiem Kameraspeicher. Ist nicht ausreichend freier Speicher vorhanden, so werden zumindest Teile alter Sequenzen überschrieben.
Von einer alten Sequenz sind also eventuell nur noch einige Bilder gültig
Die Funktion *is_TransferImage* erwartet als Parameter die Sequenz Id einer gültigen Sequenz und die Nummer des zu übertragenden Bildes innerhalb dieser Sequenz. Ist das Bild oder die gesamte Sequenz inzwischen ungültig, gibt die Funktion einen entsprechenden Fehler zurück.

**Beispiel:**
Es wurde bereits eine Sequenz (A) mit 5 Bildern aufgenommen. Eine zweite Sequenz (B) passt noch komplett in den Speicher, ohne Überschreibungen vornehmen zu müssen.
Wird nun eine dritte Sequenz aufgenommen, deren 3 Bilder zusammen größer sind als der noch zur Verfügung stehende Speicher, so wird der Speicher wieder vom Anfang an beschrieben, falls ein Bild nicht mehr komplett in den noch freien Speicher passt (einzelne Bilder werden also nicht umgebrochen, es bleibt ein kleiner Rest ungenutzt).
Eventuell in diesen Speicherstellen stehende Bilder werden überschrieben, im Beispiel ragt das Bild C3 in den Speicherbereich von A3, womit das ganze Bild A3 ungültig wird. Sequenz A besteht nunmehr aus nur noch 2 Bildern.

Abb. 10: Anhängender Speichermodus

**Timing im Memoryboard-Betrieb**
Die Daten eines aufgenommenen Bildes werden im Normalbetrieb direkt vom Sensor über die USB-Schnittstelle an den Rechner geschickt.
Bei der Verwendung des Memoryboards verhält sich dies anders. Im Memoryboard-Betrieb befindet sich der Sensor im Triggermodus. Sobald das Triggersignal anliegt, wird gemäß der Voreinstellungen 1 Bild oder eine Folge von Bildern aufgenommen und im Memoryboard gespeichert. Nachdem das letzte Bild gespeichert wurde, kann die Übertragung der gespeicherten Bilder über die USB-Schnittstelle zum Rechner erfolgen. Während der Übertragung der Daten können keine neuen Bilder aufgenommen und gespeichert werden.

*Abb. 11: Ablauf Direct mode / Memory mode*

Das nachfolgende Diagramm zeigt die zeitlichen Abläufe bei der Verwendung des Memoryboards.



*Abb. 12: Timing Diagramm Memoryboard*

 

## 3.13. Gültigkeit in Darstellungsmodi

Einige der Funktionen sind für alle Darstellungsmodi gültig, andere arbeiten nur in Verbindung mit den DirectDraw Darstellungsmodi (DD-Modi). In der nachfolgenden Tabelle sind die Gültigkeiten der jeweiligen Funktionen aufgelistet.

| Funktion | Bitmap | DD-BackBuffer Surface | DD-Overlay Surface |
|---|---|---|---|
| is_AddToSequence | ☒ | | |
| is_AllocImageMem | ☒ | ✸ | ✸ |
| is_CaptureVideo | ☒ | ☒ | ☒ |
| is_ClearSequence | ☒ | | |
| is_ConvertImage | ☒ | | |
| is_CopyImageMem | ☒ | ✸ | ✸ |
| is_CopyImageMemLines | ☒ | ✸ | ✸ |
| is_DisableDDOverlay | | ☒ | |
| is_EnableDDOverlay | | ☒ | |
| is_FreeImageMem | ☒ | ✸ | ✸ |
| is_FreezeVideo | ☒ | ☒ | ☒ |
| is_GetActiveImageMem | ☒ | | |
| is_GetActSeqBuf | ☒ | | |
| is_GetDC | | ☒ | ☒ |
| is_GetDDOvlSurface | | ☒ | |
| is_GetImageHistogram | ☒ | ✸ | ✸ |
| is_GetImageMem | ☒ | ☒ | ☒ |
| is_GetImageMemPitch | ☒ | ✸ | ✸ |
| is_GetVsyncCount | ☒ | ☒ | ☒ |
| is_HasVideoStarted | ☒ | ☒ | ☒ |
| is_HideDDOverlay | | ☒ | |
| is_InquireImageMem | ☒ | ✸ | ✸ |
| is_IsVideoFinish | ☒ | ☒ | ☒ |
| is_LoadImage | ☒ | | |
| is_LoadImageMem | ☒ | | |
| is_LockDDMem | | ☒ | ☒ |
| is_LockDDOverlayMem | | ☒ | |
| is_LockSeqBuf | ☒ | | |
| is_PrepareStealVideo | ☒ | ☒ | ☒ |
| is_ReleaseDC | | ☒ | ☒ |
| is_RenderBitmap | ☒ | | |
| is_SaveImage | ☒ | ☒ | ☒ |
| is_SaveImageEX | ☒ | ☒ | ☒ |
| is_SaveImageMem | ☒ | ✸ | ✸ |
| is_SaveImageMemEx | ☒ | ✸ | ✸ |
| is_SetAllocatedImageMem | ☒ | ✸ | ✸ |
| is_SetBayerConversion | ☒ | ☒ | ☒ |
| is_SetBinning | ☒ | ☒ | ☒ |
| is_SetColorMode | ☒ | ☒ | ☒ |
| is_SetConvertParam | ☒ | | |
| is_SetDisplayMode | ☒ | ☒ | ☒ |
| is_SetDisplayPos | ☒ | | |
| is_SetHwnd | | ☒ | ☒ |
| is_SetImageMem | ☒ | ✸ | ✸ |
| is_SetImagePos | ☒ | ☒ | ☒ |

 

| | | | |
|---|---|---|---|
| is_SetImageSize | ☒ | ☒ | ☒ |
| is_SetKeyColor | | | ☒ |
| is_SetRopEffect | ☒ | ☒ | ☒ |
| is_SetSaturation | | | ☒ |
| is_SetSubSampling | ☒ | ☒ | ☒ |
| is_ShowDDOverlay | | ☒ | |
| is_StealVideo | | ☒ | ☒ |
| is_StopLiveVideo | ☒ | ☒ | ☒ |
| is_UnlockDDMem | | ☒ | ☒ |
| is_UnlockDDOverlayMem | | ☒ | |
| is_UnlockSeqBuf | ☒ | | |
| is_UpdateDisplay | | ☒ | ☒ |

✸ *nur im Stealmode; Speicher in DD-Modes nicht notwendig*

*Tabelle 15: Gültigkeit der Funktionen*

DirectDraw wird unter LINUX nicht unterstützt.

## 3.14. Nicht unterstütze Funktionen

Die nachfolgend aufgeführten Funktionen sind spezielle Funktionen für die FALCON Framegrabber Familie und werden von der uEye-Kamera-Familie nicht unterstützt.

*is_GetCurrentField()*
*is_GetIRQ()*
*is_GetPciSlot()*
*is_OvlSurfaceOffWhileMove()*
*is_ScaleDDOverlay()*
*is_SetAGC()*
*is_SetCaptureMode()*
*is_SetDecimationMode()*
*is_SetDisplaySize()*
*is_SetHorFilter()*
*is_SetHue()*
*is_SetIOMask()*
*is_SetKeyOffset()*
*is_SetParentHwnd()*
*is_SetPassthrough()*
*is_SetRenderMode()*
*is_SetSync()*
*is_SetSyncLevel()*
*is_SetToggleMode()*
*is_SetUpdateMode()*
*is_SetVertFilter()*
*is_SetVideoCrossbar()*
*is_SetVideoInput()*
*is_SetVideoMode()*
*is_SetVideoSize()*
*is_ShowColorBars()*
*is_Watchdog()*
*is_WatchdogTime()*

# 4. Beschreibung der Funktionen

Zur Einbindung der uEye Kameras in eigene Programme stellt die Treiberbibliothek die in diesem Kapitel beschriebenen Funktionen und Parameter zur Verfügung. Die Funktionen sind alphabetisch sortiert und wie folgt aufgebaut:

**<Name der Funktion>**

**Syntax:**

Prototyp der Funktion aus der Header-Datei ueye.h

**Beschreibung:**

Beschreibung der Funktion mit Querverweisen auf betroffene Funktionen

**Übergabeparameter:**

Beschreibung der Funktionsparameter mit Wertebereichen

**Rückgabewert:**

Beschreibung und Wertebereich des Rückgabewerts. Liefert eine Funktion den Wert IS_NO_SUCCESS (-1) zurück, kann der Fehler mit der Funktion *is_GetError()* abgefragt werden.

Der Source Code des Beispielprogramms *uEye_demo.exe*, das die Funktionen der uEye-Bibliothek verwendet, liegt auf der Installations-CD bei. Darin werden neben der eigentlichen Initialisierung der uEye Kamera und des Zugriffs auf die Kamera auch die verschiedenen Modi der Bilddarstellung gezeigt.

## 4.1. is_AddToSequence

**Syntax:**

INT is_AddToSequence (HIDS hf, char* pcImgMem, INT nID)

**Beschreibung:**

*is_AddToSequence()* fügt einen Bildspeicher in die Liste der Bildspeicher ein, die für Ringbuffering verwendet werden. Der Bildspeicher muss zuvor mit *is_AllocImageMem()* angefordert worden sein. Bildspeicher die für das Ringbuffering verwendet werden, müssen für dieselbe Farbtiefe (Bits per Pixel) allokiert worden sein. Die Anzahl der Bildspeicher für eine Sequenz *(nID)* ist auf den Integer-Wertebereich begrenzt.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **pcMem** | Zeiger auf Bildspeicher |
| **nID** | ID des Bildspeichers |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.2. is_AllocImageMem

**Syntax**

INT is_AllocImageMem (HIDS hf, INT width, INT height, INT bitspixel, char** ppcImgMem, INT* pid)

**Beschreibung**

*is_AllocImageMem()* allokiert einen Bildspeicher für ein Bild der Breite *width*, der Höhe *height* und der Farbtiefe *bitspixel*. Die Speichergröße ist mindestens:
size = [width ∗ ((bitspixel + 1) / 8) + adjust] ∗ height
*adjust* siehe unten

Der Zeileninkrement berechnet sich zu:
line = width ∗ [(bitspixel + 1) / 8]
lineinc = line + adjust.
adjust = 0, wenn *line* ohne Rest durch 4 teilbar ist
adjust = 4 - rest(line / 4), wenn *line* nicht ohne Rest durch 4 teilbar ist

Der Zeileninkrement kann mit der Funktion *is_GetImgMemPitch()* ausgelesen werden.
Die Anfangsadresse des Speicherbereichs wird in *ppcImgMem* zurückgegeben.
*pid* enthält eine Identifikationsnummer des allokierten Speichers. Ein neu allokierter Speicher ist nicht direkt aktiv, d.h. Bilder werden nicht direkt in diesen neuen Speicher digitalisiert. Er muss zuerst mit *is_SetImageMem()* aktiv gesetzt werden.
Der zurückgegebene Zeiger muss gesichert werden und darf nicht verändert werden, da dieser für alle weiteren *ImageMem*-Funktionen benötigt wird! Die Freigabe des Speichers erfolgt mit *is_FreeImageMem().*
In den DirectDraw Modi ist das Allokieren eines Bildspeichers nicht notwendig!

> Aktuelle Betriebssysteme lagern einzelne, länger nicht benutzte Bereiche des Arbeitsspeichers auf die langsamere Festplatte aus, falls der freie physische Arbeitsspeicher knapp wird. Daher kann sich die Bildaufnahme insgesamt verlangsamen, falls mehr Bildspeicher reserviert wurde, als gleichzeitig im Arbeitsspeicher vorgehalten werden kann.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **width** | Breite des Bildes |
| **height** | Höhe des Bildes |
| **bitspixel** | Farbtiefe des Bildes (Bits pro Pixel) |
| **ppcImgMem** | Enthält dann den Zeiger auf den Speicheranfang |
| **pid** | Enthält dann die ID für diesen Speicher |

**Rückgabewert**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.3. is_CameraStatus

**Syntax:**

ULONG is_CameraStatus (HIDS hf, INT nInfo, ULONG ulValue)

**Beschreibung:**

Mit *is_CameraStatus()* können verschiedene Statusinformationen und Einstellungen abgefragt und teilweise gesetzt werden.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **nInfo** | |
| IS_FIFO_OVR_CNT | Anzahl FIFO Overruns. Wird erhöht, wenn Bilddaten bedingt durch einen überlasteten USB2.0 Bus verloren gehen. |
| IS_SEQUENCE_CNT | Wird bei *is_CaptureVideo()* auf 0 gesetzt. Bei jedem Wechsel des Sequenz-Buffers (Bildzähler) erfolgt die Erhöhung um 1. |
| IS_SEQUENCE_SIZE | Anzahl Sequenz-Buffer (nur lesen) |
| IS_EXT_TRIGGER_EVENT_CNT | Trigger Interrupt Zähler |
| IS_WAIT_TIMEOUT | Timeout für HW-Trigger (bei Verwendung von *IS_WAIT* oder *IS_DONT_WAIT*), 1ms Schritte. |
| IS_TRIGGER_MISSED | Anzahl der nicht verarbeiteten Triggersignale. Wird nach jedem Aufruf auf 0 gesetzt. |
| IS_LAST_CAPTURE_ERROR | Fehler beim Bildeinzug (nur lesen). |
| **ulValue** | |
| IS_GET_STATUS | Auslesen der unter *nInfo* angegebenen Information. |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS* oder der aktuelle Wert bei *ulValue = IS_GET_STATUS*

Nach Eintreten des Events *IS_SET_TRANSFER_FAILED* oder der Nachricht *IS_TRANSFER_FAILED* kann mit *IS_LASTCAPTURE_ERROR* der aufgetretene Fehler ausgelesen werden. Folgende Rückgabewerte sind möglich:

- *IS_SUCCESS*
- *IS_TRANSFER_ERROR*
  Bildeinzug wurde abgebrochen.
- *IS_TIMED_OUT*
  Die max. zulässige Zeit für den Bildeinzug wurde überschritten.
- *IS_NO_ACTIVE_IMAGE_MEM*
  Es ist kein Zielbildspeicher vorhanden.
- *IS_SEQUENCE_BUF_ALREADY_LOCKED*
  Der Zielspeicher ist nicht beschreibbar.
- *IS_COULD_NOT_CONVERT*
  Das aktuelle Bild konnte nicht konvertiert werden.

## 4.4. is_CaptureVideo

**Syntax:**

INT is_CaptureVideo (HIDS hf, INT Wait)

**Beschreibung:**

*is_CaptureVideo()* digitalisiert Videobilder in Echtzeit und überträgt die Bilder in einen allokierten Bildspeicher oder unter DirectDraw in die Grafikkarte. Die Bilddaten (DIB Mode) werden in den Speicher abgelegt, der mit is_AllocImageMem() angelegt und mit is_SetImageMem() als aktiver Bildspeicher bestimmt wurde. Über die Funktion is_GetImageMem() lässt sich die Speicheradresse abfragen. Wird mit Ringbuffering gearbeitet (*is_AddToSequence()*), so durchläuft die Bildaufnahme alle in die Sequenz aufgenommenen Bildspeicher in einer Endlosschleife.

Nach Aktivierung des Memorymodus mit *is_SetMemoryMode()* oder *is_MemoryFreezeVideo()* werden die mit *is_CaptureVideo()* erfassten Bilder im Kameraspeicher abgelegt. Um wieder eine Bilderfassung ohne Memorymode zu ermöglichen, muss der Memorymode mit der Funktion *is_SetMemoryMode(IS_MEMORY_MODE_DISABLE, 0)* (siehe *4.116 is_SetMemoryMode*) wieder abgeschaltet werden.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **Wait** | |
| IS_DONT_WAIT | Funktion synchronisiert die Bildaufnahme auf den nächsten VSYNC, kehrt aber sofort zurück |
| IS_WAIT | Funktion synchronisiert die Bildaufnahme auf den nächsten VSYNC und kehrt erst dann zurück |
| 10 < Wait < 32768 | Wartezeit in 10 ms-Schritten. Maximal kann 327,68 Sekunden (ca. 5 Minuten und 20 Sekunden) gewartet werden). Für 1 < Wait < 10 wird Wait = 10 gesetzt.<br>(Bsp.: Wait = 100 ⇒ 1 sec. warten) |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.5. is_ClearSequence

**Syntax:**

INT is_ClearSequence (HIDS hf)

**Beschreibung:**

*is_ClearSequence()* entfernt alle mit *is_AddToSequence()* hinzugefügten Bildspeicher aus der Sequenz-Liste. Nach *is_ClearSequence()* ist kein Bildspeicher mehr aktiv gesetzt. Um einen Bildspeicher als aktiven Bildspeicher zu setzen, muss *is_SetImageMem()* aufgerufen werden.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.6. is_ConvertImage

**Syntax:**

INT is_ConvertImage(HIDS hf, char* pcSource, INT nIDSource, char** ppcDest, INT *nIDDest, INT * reserved)

**Beschreibung:**

Wandelt ein RAW Bayer Bild in das gewünschte Format um. Wenn als Zeiger für das Ausgabebild der Wert NULL übergeben wird, dann wird intern ein neuer Speicher allokiert.

**Übergabeparameter:**

| | |
|---|---|
| **Hf** | Handle auf Kamera |
| **pcSource** | Zeiger auf das Eingangsbild |
| **nIDSource** | Speicher ID des Eingangsbildes |
| **ppcDest** | Zeiger auf das Ausgangsbild |
| **nIDDest** | Speicher ID des Ausgangsbildes |

**Rückgabewert:**

IS_SUCCESS oder IS_NO_SUCCESS

**Beispiel:**

Konvertierung eines Raw Bayer Bilds in RGB24. Der Speicher wird automatisch allokiert.

```
// Create a Raw Bayer test picture
char * pcSource;
INT nIDSource;
is_AllocImageMem (hf, 256, 256, 8, &pcSource, &nIDSource);
Int nX,nY,nBits,nPitch;
is_InquireImageMem (hf, pcSource, nIDSource, &nX, &nY, &nBits,
&nPitch);
for (int j = 0;j< nY;j++)
  for (int i = 0;i< nX;i++)
        pcSource[i + j*nPitch] = i;

INT Gamma = 120;
double rgbGains[3];

rgbGains[0] = 1.0 ; // Red gain
rgbGains[1] = 3.0 ; // Green gain
rgbGains[2] = 1.0 ; // Blue gain

char* pcDest;  // Pointer to the data of the new allocated picture
INT nIDDest;  // id of the new allocated picture

INT nRet;

// Set the conversion parameters
nRet = is_SetConvertParam(hf, TRUE, IS_SET_BAYER_CV_BETTER,
IS_SET_CM_RGB24, Gamma, rgbGains);
```

```
// Convert the picture
if (nRet == IS_SUCCESS){
  pcDest = NULL;
  is_ConvertImage(hf, pcSource, nIDSource, &pcDest, &nIDDest, 0);
}

// Free the allocated memory
is_FreeImageMem (m_hCam, pcSource, nIDSource);
is_FreeImageMem (m_hCam, pcDest, nIDDest);
```

## 4.7. is_CopyImageMem

**Syntax:**

INT is_CopyImageMem (HIDS hf, char* pcSource, INT nID, char* pcDest)

**Beschreibung:**

*is_CopyImageMem()* kopiert den Inhalt des durch *pcSource* und *nID* beschriebenen Bildspeichers in den Speicherbereich, auf dessen Anfang *pcDest* zeigt.

> Der Benutzer muss selbst dafür sorgen, dass der allokierte Speicher *pcDest* groß genug ist, um das gesamte Bild (nicht nur einen Bildausschnitt) im aktuellen Format (Bits per Pixel) zu fassen.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **pcSource** | Zeiger auf den Bildspeicher |
| **nID** | ID dieses Bildspeichers |
| **pcDest** | Zeiger auf den Zielspeicher, in den das Bild kopiert werden soll |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.8. is_CopyImageMemLines

**Syntax:**

INT is_CopyImageMemLines (HIDS hf, char* pcSource, INT nID, INT nLines, char* pcDest)

**Beschreibung:**

*is_CopyImageMemLines()* kopiert den Inhalt des durch *pcSource* und *nID* beschriebenen Bildspeichers in den Speicherbereich, auf dessen Anfang *pcDest* zeigt. Es werden nLines Zeilen kopiert.

> Der Benutzer muss selbst dafür sorgen, dass der allokierte Speicher *pcDest* groß genug ist, um die gewünschte Anzahl an Zeilen im aktuellen Format (Bits per Pixel) zu fassen.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **pcSource** | Zeiger auf den Bildspeicher |
| **nID** | ID dieses Bildspeichers |
| **nLines** | Anzahl Zeilen, die kopiert werden sollen |
| **pcDest** | Zeiger auf den Zielspeicher, in den das Bild kopiert werden soll |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.9. is_DisableDDOverlay

**Syntax:**

INT is_DisableDDOverlay (HIDS hf)

**Beschreibung:**

*is_EnableDDOverlay()* deaktiviert im DirectDraw BackBuffer-Modus den Overlay-Modus und gibt den durch das Overlay belegten Speicher wieder frei. Die Overlaydaten werden dadurch verworfen.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.10. is_DisableEvent

**Syntax:**

INT is_DisableEvent (HIDS hf, INT which)

**Beschreibung:**

Mit *is_DisableEvent()* wird das hier angegebene Ereignis gesperrt. Das Ereignis (z.B. ein Frame) tritt in der Regel nach wie vor auf, löst aber kein Event-Signal mehr aus. Das Anwendungsprogramm bekommt nach Aufruf dieser Funktion die gesperrten Ereignisse nicht mehr mit. Auf Wunsch kann mit *is_EnableEvent()* das gewünschte Ereignis wieder aktiviert werden. Siehe auch *4.53 is_InitEvent*.

**Übergabeparameter:**

Siehe *4.53 is_InitEvent*.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **hEv** | Event-Handle von der C/C++-Funktion *CreateEvent()* |
| **which** | ID welches Event initialisiert werden soll: |

Siehe *4.53 is_InitEvent*.

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

**Beispiel:**

Siehe *4.53 is_InitEvent*.

## 4.11. is_EnableAutoExit

**Syntax:**

INT is_EnableAutoExit (HIDS hf, INT nMode)

**Beschreibung:**

*is_EnableAutoExit()* aktiviert das automatische Schließen des Kamera-Handle nachdem eine Kamera während des laufenden Betriebs entfernt wurde. Beim Schließen werden sämtliche vom SDK reservierten Speicher wieder freigegeben.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **nMode** | |
| IS_ENABLE_AUTO_EXIT | Automatisches Schließen aktivieren. |
| IS_DISABLE_AUTO_EXIT | Automatisches Schließen deaktivieren. |
| IS_GET_AUTO_EXIT_ENABLED | Aktuelle Einstellung auslesen |

**Rückgabewert:**

Aktuelle Einstellung in Verbindung mit *IS_GET_AUTO_EXIT_ENABLED*, sonst *IS_SUCCESS* oder *IS_NO_SUCCESS*

# 4.12. is_EnableDDOverlay

**Syntax:**

INT is_EnableDDOverlay (HIDS hf)

**Beschreibung:**

*is_EnableDDOverlay()* aktiviert im DirectDraw BackBuffer-Modus den Live-Overlay-Modus. Im BackBuffer-Modus werden 3 nichtsichtbare Bildbuffer verwendet: BackBuffer, Overlay-Buffer, Mix-Buffer. Das Videobild wird in den BackBuffer digitalisiert. In den Overlay-Buffer können die Grafikdaten geschrieben werden. Der BackBuffer und der Overlay-Buffer werden dann zusammen in den Mix-Buffer geschrieben. Dabei werden die Overlaydaten dem Videobild überlagert. Der Mix-Buffer wird dann in den sichtbaren Bereich der VGA-Karte kopiert. Die drei Buffer besitzen jeweils die Größe: Video_X * Video_Y * Farbtiefe (in Bytes pro Pixel) Der Treiber versucht die Buffer direkt in der VGA-Karte zu allokieren, um den Hochgeschwindigkeits-Bildtransfer der VGA-Karte beim Mischen der drei Buffer auszunutzen. Können die Buffer nicht in der VGA-Karte allokiert werden, muss auf den Systemspeicher ausgewichen werden. Der Bildtransfer aus dem Systemspeicher ist jedoch langsamer oder unter Umständen (abhängig von der Grafikkarte) auch gar nicht möglich. Das Overlay wird nicht direkt eingeblendet. Es muss zuvor mit *is_ShowDDOverlay()* (siehe *4.125 is_ShowDDOverlay*) sichtbar gemacht werden. Das Overlay benutzt als Color-Key die Farbe Schwarz, so dass eine Overlay-Grafik keine schwarze Farbe enthalten kann (Behelf über eine nahezu schwarze Farbe, z.B. Dunkelblau)

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.13. is_EnableEvent

**Syntax:**

INT is_EnableEvent (HIDS hf, INT which)

**Beschreibung:**

Freigabe des eingerichteten Event-Objekts. Nach der Freigabe werden die Ereignismeldungen für das angelegte Event-Objekt zugelassen. Siehe auch *4.53 is_InitEvent*.

**Übergabeparameter:**

Siehe *4.53 is_InitEvent*.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **hEv** | Event-Handle von der C/C++-Funktion *CreateEvent()* |
| **which** | ID welches Event initialisiert werden soll: |

Siehe *4.53 is_InitEvent*.

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

**Beispiel:**

Siehe *4.53 is_InitEvent*.

 

# 4.14. is_EnableMessage

**Syntax:**

INT is_EnableMessage (HIDS hf, INT which, HWND hWnd)

**Beschreibung:**

Mit *is_EnableMessage()* können Nachrichten aktiviert oder deaktiviert (hWnd = NULL) werden, die beim Eintreten eines bestimmten Ereignisses an das Anwenderprogramm gesendet werden. Die Nachricht ist wie folgt aufgebaut:

| | |
|---|---|
| Msg: | *IS_UEYE_MESSAGE* |
| wParam: | Eingetroffenes Ereignis (siehe Tabelle) |
| lParam: | zur Message gehörender Kamera-Handle |

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **which** | ID der Meldung, die aktiviert/deaktiviert werden soll |
| IS_FRAME | Ein neues Bild steht zur Verfügung. |
| IS_SEQUENCE | Die Sequenz wurde durchlaufen. |
| IS_STEAL_VIDEO | Ein dem Overlay entzogenes Bild steht zur Verfügung. |
| IS_TRANSFER_FAILED | Beim Datentransfer trat ein Fehler auf (zu hohe Framerate, Pixelclock) |
| IS_TRIGGER | Ein Bild, dessen Aufnahme durch einen Trigger ausgelöst wurde, wurde komplett empfangen. Dies ist der frühest möglicher Zeitpunkt für eine neue Aufnahme. Das Bild muss noch das Postprocessing des Treibers durchlaufen und steht nach *IS_FRAME* zur Verarbeitung bereit. |
| IS_MEMORY_MODE_FINISH | Die Bildaufnahme in das optionale Memorymodul wurde beendet. |
| IS_DEVICE_REMOVED | Der mit *is_InitCamera* geöffnete Device wurde entfernt. |
| IS_DEVICE_RECONNECTED | Eine mit *is_InitCamera()* geöffnete und danach entfernte Kamera wurde wieder eingesteckt. |
| IS_NEW_DEVICE | Eine neue Kamera wurde eingesteckt. Unabhängig vom Kamera-Handle (*hf* wird ignoriert). |
| IS_DEVICE_REMOVAL | Eine Kamera wurde entfernt. Unabhängig vom Kamera-Handle (*hf* wird ignoriert). |
| IS_WB_FINISHED | Der automatische Weißabgleich hat seine Regelung beendet (falls *IS_SET_AUTO_WB_ONCE* angegeben wurde). |
| IS_AUTOBRIGHTNESS_FINISHED | Die automatische Helligkeitsregelung ist abgeschlossen (falls *IS_SET_AUTO_WB_ONCE* angegeben wurde). |
| **hWnd** | Anwendungsfenster, welches die Botschaft bekommt. (NULL deaktiviert die mit *which* bezeichnete Botschaft). |

**Rückgabewert:**

*IS_SUCCESS oder IS_NO_SUCCESS*

## 4.15. is_ExitCamera

**Syntax:**

INT is_ExitCamera (HIDS hf)

**Beschreibung:**

*is_ExitCamera()* meldet den aktiven Device Handle *hf* ab und gibt die durch die uEye Kamera besetzten Datenstrukturen und Speicherbereiche wieder frei. Durch den Anwender allokierte Bildspeicher (*is_AllocImageMem*) die noch nicht freigegeben wurden, werden durch *is_ExitCamera* freigegeben.

Im Allgemeinen ist das uEye SDK *thread safe*. Die API-Funktionsaufrufe erfolgen alle in *critical sections*. Auf Grund interner *DirectDraw* Strukturen empfehlen wird dringend, die folgenden Funktionen nur aus einem *Thread* heraus aufzurufen, um ein unvorhersehbares Verhalten Ihrer Applikation zu vermeiden.

- *is_InitCamera()*
- *is_SetDisplayMode()*
- *is_ExitCamera()*

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.16. is_ExitEvent

**Syntax:**

INT is_ExitEvent (HIDS hf, INT which)

**Beschreibung:**

Löschen des eingerichteten Event-Objekts. Nach dem Löschen des jeweiligen Events kann dieses nicht mehr mit *is_EnableEvent()* aktiviert werden.

**Übergabeparameter:**

Siehe *4.53 is_InitEvent*.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **which** | ID welches Event initialisiert werden soll: |

Siehe *4.53 is_InitEvent*.

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

**Beispiel:**

Siehe *4.53 is_InitEvent*.

## 4.17. is_ForceTrigger

**Syntax:**

INT is_ForceTrigger (HIDS hf)

**Beschreibung:**

Die Funktion *is_ForceTrigger()* ermöglicht es während einer Hardware-Trigger Aufnahme einen Trigger zu erzwingen um so unabhängig von einem echten Triggersignal ein Bild aufzunehmen. Diese Funktion kann nur angewandt werden, wenn die Triggeraufnahme mit dem Parameter *IS_DONT_WAIT* gestartet wurde.
Siehe auch *4.19 is_FreezeVideo* und *4.98 is_SetExternalTrigger*.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

**Beispiel:**

Trigger aktivieren und 1 sec. auf externen Trigger warten. Wurde kein Trigger ausgelöst Aufnahme mit is_ForceTrigger erzwingen.

```
HANDLE hEvent = CreateEvent(NULL, TRUE, FALSE, "");
if (hEvent != NULL)
{
  is_InitEvent(hf, m_hEvent, IS_SET_EVENT_FRAME);
  is_EnableEvent(hf, IS_SET_EVENT_FRAME);

  is_SetExternalTrigger(hf, IS_SET_TRIG_HI_LO);
  is_FreezeVideo(hf, IS_DONT_WAIT);
  if (WaitForSingleObject(m_hEvent, 1000) != WAIT_OBJECT_0)
  {
    // Noch kein Trigger empfangen, also Bildaufnahme erzwingen
    is_ForceTrigger(hf);
  }

  is_DisableEvent(hf, IS_SET_EVENT_FRAME);
  is_ExitEvent(hf, IS_SET_EVENT_FRAME);
}
```

 

## 4.18. is_FreeImageMem

**Syntax:**

INT is_FreeImageMem (HIDS hf, char* pcImgMem, INT id)

**Beschreibung:**

*is_FreeImageMem()* gibt einen allokierten Bildspeicher wieder frei. Für *pcImgMem* muss ein von *is_AllocImgMem()* stammender Zeiger übergeben werden. Alle anderen Zeiger führen zu einer Fehlermeldung! Eine wiederholte Übergabe des gleichen Zeigers führt ebenfalls zu einer Fehlermeldung!

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **pcImgMem** | Zeiger auf den Speicheranfang |
| **id** | ID für diesen Speicher |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.19. is_FreezeVideo

**Syntax:**

INT is_FreezeVideo (HIDS hf, INT Wait)

**Beschreibung:**

*is_FreezeVideo()* digitalisiert ein Bild und legt es im aktiven Bildspeicher ab. Im DirectDraw-Modus wird das Bild in den DirectDraw-Buffer digitalisiert (direkt in die VGA-Karte oder einen BackBuffer). Wird mit Ringbuffering gearbeitet, so erfolgt die Bildaufnahme nur in den 1. Bildspeicher der Sequenz.
Erfolgt nach dem Aufruf der Funktion mit dem Parameter *IS_DONT_WAIT* eine Veränderung der Kameraeinstellungen (Exposure, Pclk, AOI, ...) wird die Aufnahme abgebrochen und die neuen Kameraeinstellungen werden übernommen.
Die Bildaufnahme erfolgt getriggert, wenn der Triggermodus mit *is_SetExternalTrigger()* zuvor aktiviert wurde.

Nach Aktivierung des Memorymodus mit *is_SetMemoryMode()* oder *is_MemoryFreezeVideo()* werden die mit *is_FreezeVideo()* erfassten Bilder im Kameraspeicher abgelegt.
Um wieder eine Bilderfassung ohne Memorymode zu ermöglichen, muss der Memorymode mit der Funktion *is_SetMemoryMode(IS_MEMORY_MODE_DISABLE, 0)* (siehe 4.116 *is_SetMemoryMode*) wieder abgeschaltet werden.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **Wait** | |
| IS_WAIT | Funktion wartet bis Bild aufgenommen ist. Wenn die 4-fache Framezeit überschritten wurde, wird dies mit einem Timout quittiert. |
| IS_DONT_WAIT | Funktion kehrt sofort zurück |
| 10 <= Wait < 21474836 | Wartezeit in 10 ms-Schritten. Maximal kann 214748,36 Sekunden gewartet werden. Dies entspricht ca. 59,5 Stunden. Für 1 < Wait < 10 wird Wait = 10 gesetzt. (Bsp.: Wait = 100 ⇒ 1 sec warten) |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

**Beispiel:**

Triggermodus einschalten, *High-Active* Blitzmodus setzen und Bild aufnhemen.

```
is_SetExternalTrigger(hf, IS_SET_TRIG_SOFTWARE);
is_SetFlashStrobe(hf, IS_SET_FLASH_HI_ACTIVE);
is_FreezeVideo(hf, IS_WAIT);
```

## 4.20. is_GetActiveImageMem

**Syntax:**

INT is_GetActiveImageMem (HIDS hf, char** ppcMem, INT* pnID)

**Beschreibung:**

*is_GetActiveImageMem()* gibt den Zeiger auf den Anfang und die ID-Nummer des aktiven Bildspeichers zurück. Falls ein DirectDraw Modus aktiv ist, aber trotzdem Bildspeicher allokiert wurden, so wird mit dieser Funktion der Zeiger und die ID des mit *is_SetImageMem()* aktiv gesetzten Bildspeichers zurückgegeben (in diesen Bildspeicher wird im DirectDraw Modus jedoch nicht digitalisiert!) (Vergleiche auch mit *is_GetImgMem()*).

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **ppcMem** | Enthält dann den Zeiger auf den Anfang des Bildspeichers |
| **pnID** | Enthält dann die Identifikationsnummer des Bildspeichers |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.21. is_GetActSeqBuf

**Syntax:**

INT is_GetActSeqBuf (HIDS hf, INT* pnNum, char** ppcMem, char** ppcMemLast);

**Beschreibung:**

Mit *is_GetActSeqBuf()* kann der Bildspeicher ermittelt werden, in den gerade die Bildaufnahme erfolgt (*ppcMem*), bzw. den Bildspeicher der als letztes für die Bildaufnahme verwendet wurde (*ppcMemLast*). Diese Funktion ist nur verfügbar, wenn ein Ringbuffering aktiv ist. Wenn die Bildaufnahme für ein Ringbuffering gestartet wurde, so liefert *is_GetActSeqBuf()* in *pnNum* solange den Wert 0 zurück, bis die Bildaufnahme in den ersten Bildspeicher der Sequenz gestartet wurde. Sonst enthält *pnNum* die Nummer des Sequenz-Bildspeichers, in den gerade die Bildaufnahme erfolgt. Die Nummer ist nicht die ID des Bildspeichers, die von *is_AllocImageMem()* vergeben wurde, sondern die laufende Nummer in der Reihenfolge der Anmeldung über *is_AddToSequence()*.

**Übergabeparameter:**

| | | |
|---|---|---|
| **hf** | Handle auf Kamera | |
| **pnNum** | Enthält dann die Nummer des Bildspeichers in den gerade die Bildaufnahme erfolgt. | |
| | 0: | Bildaufnahme in den 1. Bildspeicher wurde noch nicht gestartet |
| | 1...max: | Bildaufnahme in den Sequenz-Bildspeicher N wurde gestartet |
| **ppcMem** | Enthält dann die Startadresse des Bildspeichers in den gerade die Bildaufnahme erfolgt | |
| **ppcMemLast** | Enthält dann die Startadresse des Bildspeichers der zuletzt für die Bildaufnahme verwendet wurde | |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.22. is_GetAutoInfo

**Syntax:**

INT is_GetAutoInfo (HIDS hf, PUEYE_AUTO_INFO info)

**Beschreibung:**

Mit der Funktion *is_GetAutoInfo()* können Statusinformationen der Autofunktionalität abgefragt werden. Die Informationen stehen in der Struktur *UEYE_AUTO_INFO* zur Verfügung.

Die in der Struktur *UEYE_AUTO_INFO* gelieferten Statusinformationen sind nur gültig, wenn mindestens eine Autofunktionalität aktiviert wurde.

**UEYE_AUTO_INFO**

| | | |
|---|---|---|
| **INT** | AutoAbility | 0x01: AutoShutter möglich (AC_SHUTTER) |
| | | 0x02: AutoGain möglich (AC_GAIN) |
| | | 0x03: AutoGain und AutoShutter möglich |
| | | 0x04: AutoWhiteBalance möglich (AC_WHITEBAL) |
| | | 0x07: Alle Autofunktionen möglich |
| **AUTO_BRIGHT_STATUS** | sBrightCtrlStatus | Siehe AUTO_BRIGHT_STATUS |
| **AUTO_WB_STATUS** | sWBCtrlStatus | Siehe AUTO_WB_STATUS |
| **DWORD** | reserviert | Reservierter Platz für Erweiterungen |

In der Struktur UEYE_AUTO_INFO werden die Strukturen AUTO_BRIGHT_STATUS und AUTO_WB_STATUS verwendet.

**AUTO_BRIGHT_STATUS**

| | | |
|---|---|---|
| **INT** | curValue | Akuteller mittlerer Grauwert (Ist) |
| **INT** | curError | Aktuelle Regelabweichung (Error) |
| **INT** | curController | Aktueller Helligkeitsregler |
| | | 0x01: AC_SHUTTER |
| | | 0x02: AC_GAIN |
| **INT** | curCtrlStatus | Aktueller Regler Status |
| | | 0x01: ACS_ADJUSTING |
| | | 0x02: ACS_FINISHED |
| | | 0x04: ACS_DISABLED |

**AUTO_WB_STATUS**

| | | |
|---|---|---|
| **INT** | curController | Akuteller Weißabgleichsregler |
| | | 0x08: AC_WB_RED_CHANNEL |
| | | 0x10: AC_WB_GREEN_CHANNEL |
| | | 0x20: AC_WB_BLUE_CHANNEL |
| **AUTO_WB_CHANNEL_STATUS** | RedChannel | Siehe AUTO_WB_CHANNEL_STATUS |
| **AUTO_WB_CHANNEL_STATUS** | GreenChannel | Siehe AUTO_WB_CHANNEL_STATUS |
| **AUTO_WB_CHANNEL_STATUS** | BlueChannel | Siehe AUTO_WB_CHANNEL_STATUS |

Nachfolgend wird die Struktur AUTO_WB_CHANNEL_STATUS erläutert, welche in der Struktur AUTO_WB_STATUS zum Einsatz kommt.

**AUTO_WB_CHANNEL_STATUS**

| | | |
|---|---|---|
| **INT** | curValue | Akuteller mittlerer Grauwert (Ist) |
| **INT** | curError | Aktuelle Regelabweichung (Error) |
| **INT** | curCtrlStatus | Aktueller Reglerstatus<br>0x01: ACS_ADJUSTING<br>0x02: ACS_FINISHED<br>0x04: ACS_DISABLED |

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **info** | Struktur UEYE_AUTO_INFO (siehe oben) |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

**Beispiel:**

Auto Info abragen:

```
UEYE_AUTO_INFO autoinfo;
Int ret = is_GetAutoInfo(m_hCam, &autoinfo);
```

# 4.23. is_GetBusSpeed

**Syntax:**

INT is_GetBusSpeed (HIDS hf)

**Beschreibung:**

Mit der Funktion *is_GetBusSpeed()* kann abgefragt werden, ob eine Kamera an einen USB 2.0 Hostcontroller angeschlossen ist.

Bei Übergabe des Wertes Null(0) für den Kamera Handle wird überprüft, ob ein USB 2.0 Kontroller im System vorhanden ist.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |

**Rückgabewert:**

| | |
|---|---|
| IS_SUCCESS | |
| IS_NO_SUCCESS | kein USB 2.0 Kontroller vorhanden (hf=0) |
| IS_USB_10 | Der Kontroller, an dem die Kamera angeschlossen wurde, unterstützt kein USB 2.0. |
| IS_USB_20 | Kamera ist an einem USB 2.0 Kontroller angeschlossen |

## 4.24. is_GetCameraInfo

**Syntax:**

INT is_GetCameraInfo (HIDS hf, PCAMINFO pInfo)

**Beschreibung:**

Die Funktion *is_GetCameraInfo()* liest die fest im EEPROM hinterlegten Daten aus und schreibt diese in die Datenstruktur auf die *pInfo* zeigt. Die Datenstruktur ist in Kapitel *2.3 CAMINFO Datenstruktur des EEPROMS* beschrieben.

Das Lesen und Schreiben eigener Daten in und aus dem EEPROM wird über die Funktionen *is_ReadEEPROM()* und *is_WriteEEPROM()* durchgeführt.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **pInfo** | Zeiger auf eine Datenstruktur CAMINFO |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.25. is_GetCameraList

**Syntax:**

INT is_GetCameraList (PUEYE_CAMERA_LIST pucl)

**Beschreibung:**

Mit der Funktion *is_GetCameraList()* können Informationen über die angeschlossenen Kameras abgefragt werden. In den nachfolgenden Tabellen werden die verwendeten Strukturen erläutert.

UEYE_CAMERA_LIST

| | | |
|---|---|---|
| ULONG | dwCount | Anzahl der am System angeschlossenen Kameras |
| UEYE_CAMERA_INFO | uci[1] | Platzhalter für 1 .. n Strukturen UEYE_CAMERA_INFO |

UEYE_CAMERA_INFO

| | | |
|---|---|---|
| DWORD | dwCameraID | Benutzerdefinierbare Kamera ID |
| DWORD | dwDeviceID | Systeminterne Geräte ID |
| DWORD | dwSensorID | Sensor ID |
| DWORD | dwInUse | 1 = Kamer in Verwendung<br>0 = Kamera nicht in Verwendung |
| Char | SerNo[16] | Seriennummer der Kamera |
| Char | Model[16] | Kameramodell |
| DWORD | dwReserved[16] | Für spätere Verwendung reserviert |

**Übergabeparameter:**

| | |
|---|---|
| **pucl** | Struktur UEYE_CAMERA_LIST |

**Rückgabewert:**

*IS_SUCCESS, IS_ACCESS_VIOLATION* (zu wenig Speicher allokiert) oder *IS_CANT_OPEN_DEVICE bzw. IS_IO_REQUEST_FAILED* (Kommunikation mit Treiber fehlgeschlagen)

**Beispiel:**

```
PUEYE_CAMERA_LIST pucl = new UEYE_CAMERA_LIST;
//first request number of cameras to determine the array size
//within the UEYE_CAMERA_LIST structure
pucl->dwCount = 0;
if (is_GetCameraList (pucl) == IS_SUCCESS){
  //get number of cameras
  DWORD dwCameraCount = pucl->dwCount;
  delete pucl;
  //reallocate the required list size
  pucl = (PUEYE_CAMERA_LIST) new char [sizeof (DWORD) + dwCameraCount
  * sizeof (UEYE_CAMERA_INFO)];
  pucl->dwCount = dwCameraCount;
  //let the DLL fill in the camera info
  if (is_GetCameraList(pucl) == IS_SUCCESS){
    for (int iCamera = 0; iCamera < (int) pucl->dwCount; iCamera++){
```

 

```
            //process camera info
            printf("Camera %i Id: %d", iCamera,
            pucl->uci[iCamera].dwCameraID);
        }
    }
}
```

## 4.26. is_GetCameraType

**Syntax:**

INT is_GetCameraType (HIDS hf)

**Beschreibung:**

*is_GetCameraType()* liefert als Ergebnis den Typ der Kamera-Familie zurück. Bei der uEye Kamera Familie wird immer *IS_CAMERA_TYPE_UEYE_USB* zurückgegeben.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |

**Rückgabewert:**

IS_CAMERA_TYPE_UEYE_USB

## 4.27. is_GetColorDepth

**Syntax:**

INT is_GetColorDepth(HIDS hf, INT* pnCol, INT* pnColMode)

**Beschreibung:**

*is_GetColorDepth()* ermittelt die aktuelle Farbeinstellung der VGA-Karte und gibt die Bittiefe (*pnCol*) und den zu *pnCol* passenden uEye Farbmodus (*pnColMode*) zurück. Der Farbmodus kann direkt der Funktion *is_SetColorMode()* übergeben werden.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **PnCol** | Liefert die Bittiefe der Farbeinstellung zurück |
| | 8 bei 256 Farben |
| | 15 bei 32768 Farben (5:5:5 Modus) |
| | 16 bei 65536 Farben (5:6:5 Modus) |
| | 24 bei 16777216 Farben (8:8:8 Modus) |
| | 32 bei 16777216 Farben (0:8:8:8 Modus) |
| **pnColMode** | Liefert den zu pnCol entsprechenden uEye Farbmodus zurück |
| | IS_SET_CM_Y8 bei pnCol = 8 |
| | IS_SET_CM_RGB15 bei pnCol = 15 |
| | IS_SET_CM_RGB16 bei pnCol = 16 |
| | IS_SET_CM_RGB24 bei pnCol = 24 |
| | IS_SET_CM_RGB32 bei pnCol = 32 |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.28. is_GetDC

**Syntax:**

INT is_GetDC (HIDS hf, HDC* phDC)

**Beschreibung:**

*is_GetDC()* gibt im DirectDraw BackBuffer-Modus das Device-Context-Handle des Overlaybuffers zurück. Mit diesem Handle kann über die Windows-GDI Funktionen auf das Overlay zugegriffen werden. Es stehen somit alle Grafikbefehle wie Line, Circle, Rectangle, TextOut,... Funktionen von Windows zur Verfügung. Das Device-Context-Handle muss sobald als möglich mit der Funktion *is_ReleaseDC()* freigegeben werden. Innerhalb eines *GetDC - ReleaseDC* Blocks erfolgt KEIN Update des Overlaybuffers auf dem Bildschirm

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **phDC** | Zeiger auf Variable, die das Device-Kontext-Handle aufnimmt |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.29. is_GetDDOvlSurface

**Syntax:**

INT is_GetDDOvlSurface (HIDS hf, LPDIRECTDRAWSURFACE* ppDDSurf)

**Beschreibung:**

*is_GetDDOvlSurface()* gibt im DirectDraw BackBuffer-Modus den Zeiger auf das interne DirectDraw Surface zurück. Damit können die Funktionen des IDirectDraw-Surface Interface verwendet werden.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **ppDDSurf** | Enthält dann den Zeiger auf das IDirectDraw Surface Interface |

**Rückgabeparameter:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.30. is_GetDLLVersion

**Syntax:**

INT is_GetDLLVersion ()

**Beschreibung:**

Gibt die Version der ueye_api.dll zurück.
Der Rückgabewert enthält die Versionsnummer in folgender Kodierung:

| | |
|---|---|
| Bits 31-24: | major version number |
| Bits 16-23: | minor version number |
| Bits 15-0: | build version number |

**Übergabeparameter:**

<keine>

**Rückgabewert:**

Versionsnummer

## 4.31. is_GetError

**Syntax:**

INT is_GetError (HIDS hf, INT* pErr, char** ppcErr)

**Beschreibung:**

*is_GetError()* fragt den letzten aufgetretenen Fehler ab und liefert Fehlercode und Fehlermeldung zurück. Die letzte Fehlermeldung wird nicht gelöscht, aber durch neu eintretende Fehler überschrieben.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **PErr** | Zeiger auf Variable, die dann den Fehlercode enthält |
| **PpcErr** | Zeiger auf String, der dann den Fehlertext enthält |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.32. is_GetExposureRange

**Syntax:**

INT is_GetExposureRange (HIDS hf, double *min,double *max, double *intervall)

**Beschreibung:**

Mit *is_GetExposureRange()* können die für das momentan eingestellte Timing (Pixeltakt, Framerate) möglichen Exposure Werte in Millisekunden abgefragt werden. Die möglichen Zeiten liegen zwischen *min* und *max* und können in *intervall* großen Schritten eingestellt werden.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **min** | Enthält die minimal mögliche Exposure Zeit |
| **max** | Enthält die maximal mögliche Exposure Zeit |
| **intervall** | Enthält die Schrittweite, mit der die Bilddauer geändert werden kann. |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.33. is_GetFramesPerSecond

**Syntax:**

INT is_GetFramesPerSecond (HIDS hf, double *dblFPS)

**Beschreibung:**

*is_GetFramesPerSecond()* gibt im Livetrieb (siehe *4.4 is_CaptureVideo*) die Anzahl der tatsächlich eingelesenen Bilder pro Sekunde zurück.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **dblFPS** | Framerate |

**Rückgabewert:**

*IS_SUCCESS*, *IS_NO_SUCCESS*

## 4.34. is_GetFrameTimeRange

**Syntax:**

INT is_GetFrameTimeRange (HIDS hf, double *min, double *max, double *intervall)

**Beschreibung:**

Mit *is_GetFrameTimeRange()* können die möglichen Einstellungen der Framerate ausgelesen werden, die für die momentanen Einstellungen des Pixeltaktes möglich sind. Die zurückgegeben Werte geben die minimal und maximal mögliche Dauer eines Bildes in Sekunden an. Die mögliche Framedauer kann zwischen *min* und *max* in *intervall* Schritten eingestellt werden.

$$fps(\min) = \frac{1}{\max}$$

$$fps(\max) = \frac{1}{\min}$$

$$fps(n) = \frac{1}{\min + m}$$

m = n*intervall

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **min** | Enthält die minimal mögliche Dauer eines Bildes |
| **max** | Enthält die maximal mögliche Dauer eines Bildes |
| **intervall** | Enthält die Schrittweite, mit der die Bilddauer geändert werden kann. |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.35. is_GetGlobalFlashDelays

**Syntax:**

INT is_GetGlobalFlashDelays (HIDS hf, ULONG *pulDelay, ULONG *pulDuration)

**Beschreibung:**

Mit der Funktion *is_GetGlobalFlashDelays()* können die benötigten Zeiten ermittelt werden um eine globale Blitzfunktion für *Rolling Shutter Kameras* zu realisieren. Somit kann eine *Rolling Shutter Kamera* als *Global Shutter Kamera* betrieben werden, wenn die zu erfassende Bild-szene in der Blitzpause zwischen zwei Bildern dunkel ist.
Wird die Belichtungszeit zu kurz eingestellt, so dass kein globaler Blitz mehr möglich ist, wird *IS_NO_SUCCESS* zurückgegeben.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **ulDelay** | Zeit um die der Blitz verzögert wird (in µs) |
| **ulDuration** | Zeit in der der Blitz eingeschaltet wird (in µs) |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS.*

## 4.36. is_GetImageHistogram

**Syntax:**

INT is_GetImageHistogram (HIDS hf, int nID, INT ColorMode, DWORD* pHistoMem)

**Beschreibung:**

*is_GetImageHistogram()* berechnet das Histogramm des übergebenen Bildes. Dabei werden die Farbformate RGB32, RGB24, RGB16, RGB15, Raw Bayer und Y8 unterstützt.

**Übergabeparameter:**

| | |
|---|---|
| **Hf** | Handle auf Kamera |
| **nID** | Memory ID |
| **ColorMode** | Farbmodus des Bildes mit Memory ID *nID* |
| IS_SET_CM_RGB32 | DWORD Array [256*3] |
| IS_SET_CM_RGB24 | DWORD Array [256*3] |
| IS_SET_CM_RGB16 | DWORD Array [256*3] |
| IS_SET_CM_RGB15 | DWORD Array [256*3] |
| IS_SET_CM_BAYER | DWORD Array [256*3] |
| IS_SET_CM_Y8 | DWORD Array [256] |
| **pHistoMem** | Zeiger auf ein DWORD Array |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*
*IS_NULL_POINTER* invalid array
*IS_INVALID_COLOR_FORMAT nicht unterstütztes Farbformat*
*IS_INVALID_PARAMETER* unbekannter Parameter ColorMode

**Beispiel:**

RGB Testbild erzeugen

```
char * pcSource;
INT nIDSource;
is_AllocImageMem (hf, 256, 256, 24, &pcSource, &nIDSource);
Int nX,nY,nBits,nPitch;
is_InquireImageMem (hf,pcSource,nIDSource,&nX,&nY,&nBits,&nPitch);
for (int j = 0;j< nY;j++){
  for (int i = 0;i< nX*3;i+=3){
      pcSource[i + j*nPitch] = 0; // Blue pixels
      pcSource[i + j*nPitch + 1] = i/3; // Green pixels
      pcSource[i + j*nPitch + 2] = 255; // Red pixels
  }
}
// memory for the rgb histogram
DWORD bgrBuffer [256*3];
//Assign a pointer to each color histogram
DWORD * pBlueHisto  = bgrBuffer;
DWORD *pGreenHisto = bgrBuffer + 256;
DWORD * pRedHisto = bgrBuffer + 512;
is_GetImageHistogram (hf, nIDSource, IS_SET_CM_RGB24, bgrBuffer);
is_FreeImageMem (hf, pcSource, nIDSource);
```

## 4.37. is_GetImageMem

**Syntax:**

INT is_GetImageMem (HIDS hf, VOID** pMem)

**Beschreibung:**

*is_GetImageMem()* gibt den Zeiger auf den Anfang des aktiven Bildspeichers zurück. Im DirectDraw-Modus wird der Zeiger auf den BackBuffer bzw. der Zeiger auf den sichtbaren Bereich (DirectDraw Primary Surface- Modus) zurückgegeben.

> Im DirectDraw Primary-Modus verändert sich der Adresszeiger wenn das Ausgabefenster verschoben wird. (siehe auch *4.61 is_LockDDMem*). Wenn mit Ringbuffering gearbeitet wird (4.1 is_AddToSequence), liefert *is_GetImageMem()* den zuvor aktiven Bildspeicher zurück!

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **pMem** | Zeiger auf den Anfang des Bildspeichers |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

 

## 4.38. is_GetImageMemPitch

**Syntax:**

INT is_GetImageMemPitch (HIDS hf, INT* pPitch)

**Beschreibung:**

*is_GetImageMemPitch()* gibt den Zeileninkrement in Bytes zurück. Der Zeileninkrement ist die Anzahl Bytes vom Beginn einer Zeile bis zum Beginn der nächsten Zeile. Das Zeileninkrement kann u.U. größer sein als aus den beim Aufruf von *is_AllocImageMem()* übergebenen Parametern ersichtlich ist (Das Zeileninkrement ist immer eine durch 4 teilbare Anzahl Bytes, siehe 4.2 is_AllocImageMem);
Der Zeileninkrement berechnet sich zu:
line = width ∗ [(bitspixel + 1) / 8]
lineinc = line + adjust.
adjust = 0 wenn *line* ohne Rest durch 4 teilbar ist
adjust = 4 - rest(line / 4) wenn *line* nicht ohne Rest durch 4 teilbar ist

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **pPitch** | Zeiger auf Variable die dann den Zeileninkrement enthält |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.39. is_GetLastMemorySequence

**Syntax:**

INT is_GetLastMemorySequence (HIDS hf, INT *pID)

**Beschreibung:**

Die Funktion *is_GetLastMemorySequence()* gibt die ID der zuletzt aufgenommenen Sequenz im Memoryboard zurück. Dieser Parameter kann dann in Verbindung mit der Funktion *is_TransferImage()* verwendet werden um Bilder aus dem Kameraspeicher auszulesen.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **pID** | Gibt die ID der zuletzt aufgenommenen Sequenz im Speicher zurück. |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

 

## 4.40. is_GetMemorySequenceWindow

**Syntax:**

INT is_GetMemorySequenceWindow (HIDS hf, INT nID, INT *left, INT *top, INT *right, INT *bottom)

**Beschreibung:**

Mit der Funktion *is_GetMemorySequenceWindow()* kann die Fenstergröße zu einer angegebenen Memoryboard Sequenz abgefragt werden. Als Parameter wird die gewünschte Sequenz ID benötigt.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **nID** | Sequenz ID zu der die Fensterkoordinaten abgefragt werden. |
| **left** | Gibt die Anfangs-Spalte zurück. |
| **top** | Gibt die Anfangs-Zeile zurück |
| **right** | Gibt die End-Spalte zurück |
| **bottom** | Gibt die End-Zeile zurück |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.41. is_GetNumberOfCameras

**Syntax:**

INT is_GetNumberOfCameras (INT* pnNumCams)

**Beschreibung:**

*is_GetNumberOfCameras()* gibt die Anzahl der am PC angeschlossenen uEye Kameras zurück

**Übergabeparameter:**

| | |
|---|---|
| pnNumCams | Gibt die Anzahl angeschlossener Kameras zurück. |

**Rückgabewert:**

*IS_SUCCESS*

## 4.42. is_GetNumberOfMemoryImages

**Syntax:**

INT is_GetNumberOfMemoryImages (HIDS hf, INT nID, INT *pnCount)

**Beschreibung:**

Die Funktion *is_GetNumberOfMemoryImages()* gibt die Anzahl an gültigen Bildern zurück, die sich innerhalb der angegebenen Sequenz-ID im Kameraspeicher befinden. Diese Anzahl kann sich aufgrund von Überschreibungen von der ursprünglich aufgenommenen Anzahl an Bildern unterscheiden.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **nID** | Gibt die ID an, zu der die Anzahl der vorhandenen Bilder zurückgegeben werden soll. |
| **pnCount** | Dieser Parameter gibt die Anzahl der sich in der Sequenz befindenden Bilder zurück. |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.43. is_GetOsVersion

**Syntax:**

INT is_GetOsVersion ()

**Beschreibung**

*is_GetOsVersion()* gibt den Typ des Betriebssystems zurück, mit dem zur Laufzeit gearbeitet wird.

**Übergabeparameter:**

<keine>

**Rückgabewert:**

*IS_OS_WIN2000, IS_OS_WINXP, IS_OS_WINSERVER2003*

## 4.44. is_GetPixelClockRange

**Syntax:**

INT is_GetPixelClockRange (HIDS hf, INT *pnMin, INT *pnMax)

**Beschreibung:**

*is_GetPixelClockRange()* gibt den einstellbaren Bereich für den Pixeltakt zurück.

Je nach Kameramodell und Betriebsmodus können die Grenzwerte für den Pixeltakt variieren. So erhöht sich bei einer UI-1220x der minimale Pixeltaktbei aktiviertem 4x-Binning (horizontal) von 8 MHz auf 16 MHzund bei einer UI-141x von 5MHz auf 10MHz bei aktiviertem 2x-Sampling (horizontal).

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **pnMin** | Gibt den unteren Grenzwert zurück. |
| **pnMax** | Gibt den oberer Grenzwert zurück. |

**Rückgabewert:**

IS_SUCCESS oder IS_NO_SUCCESS

## 4.45. is_GetRevisionInfo

**Syntax:**

INT is_GetRevisionInfo(HCAM hf, PREVISIONINFO prevInfo)

**Beschreibung:**

Die Funktion *is_GetRevisionInfo()* liest die Revisionsinformationen der einzelnen uEye Komponenten aus. Die Revisions-Struktur ist wie folgt aufgebaut:

| | |
|---|---|
| **Size** | Strukturgröße |
| **Sensor** | Hardware Sensorrevision |
| **Cypress** | |
| Highbyte: | 1 = Typ FX2<br>2 = Typ FX2LP |
| Lowbyte: | Cypress revision |
| **Blackfin** | DSP revision |
| **DSPFirmware** | Blackfin Firmware Version |
| **USB_Board** | USB Board Revision |
| **Sensor_Board** | Sensor Board Revision |
| **Processing_Board** | Processing Board Revision |
| **Memory_Board** | Memory Board Revision |
| **Housing** | Gehäusevariante |
| **Filter** | Eingebauter Filtertyp |
| **Timing_Board** | Timing Board Revision |
| **Product** | Produktrevision |
| **Reserved** | Für spätere Verwendung reserviert |

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **prevInfo** | Zeiger auf Revisions-Informations-Struktur |

**Rückgabewert:**

IS_SUCCESS, IS_NO_SUCCESS

# 4.46. is_GetSensorInfo

**Syntax:**

INT is_GetSensorInfo (HIDS hf, PSENSORINFO pInfo)

**Beschreibung:**

Mit Hilfe der Funktion *is_GetSensorInfo()* können Informationen über den verwendeten Sensor abgefragt werden. Die in der Struktur *SENSORINFO* enthaltenen Informationen sind in der folgenden Tabelle aufgelistet:

WORD SensorID

| Sensortyp | hex | dec |
|---|---|---|
| **CMOS** | | |
| IS_SENSOR_INVALID | 0 | 0 |
| IS_SENSOR_UI141X_M | 1 | 1 |
| IS_SENSOR_UI141X_C | 2 | 2 |
| IS_SENSOR_UI144X_M | 3 | 3 |
| IS_SENSOR_UI144X_C | 4 | 4 |
| IS_SENSOR_UI145X_C | 8 | 8 |
| IS_SENSOR_UI146X_C | A | 10 |
| IS_SENSOR_UI121X_M | 10 | 16 |
| IS_SENSOR_UI121X_C | 11 | 17 |
| IS_SENSOR_UI122X_M | 12 | 18 |
| IS_SENSOR_UI122X_C | 13 | 19 |
| IS_SENSOR_UI164X_C | 20 | 32 |
| IS_SENSOR_UI154X_M | 30 | 48 |
| IS_SENSOR_UI154X_C | 31 | 49 |
| IS_SENSOR_UI1543_M | 32 | 50 |
| IS_SENSOR_UI1543_C | 33 | 51 |
| **CCD** | | |
| IS_SENSOR_UI223X_M | 80 | 128 |
| IS_SENSOR_UI223X_C | 81 | 129 |
| IS_SENSOR_UI241X_C | 82 | 130 |
| IS_SENSOR_UI241X_C | 83 | 131 |
| IS_SENSOR_UI221X_M | 88 | 136 |
| IS_SENSOR_UI221X_C | 89 | 137 |
| IS_SENSOR_UI231X_M | 90 | 144 |
| IS_SENSOR_UI231X_C | 91 | 145 |
| IS_SENSOR_UI222x_M | 92 | 146 |
| IS_SENSOR_UI222x_C | 93 | 147 |
| IS_SENSOR_UI233x_M | 94 | 148 |
| IS_SENSOR_UI233x_C | 95 | 149 |
| IS_SENSOR_UI224x_M | 96 | 150 |
| IS_SENSOR_UI224x_C | 97 | 151 |
| IS_SENSOR_UI225x_M | 98 | 152 |
| IS_SENSOR_UI225x_C | 99 | 153 |

| | | |
|---|---|---|
| Char | strSensorName[32] | Sensor Name z.B. "UI141x_M" |
| Char | nColorMode | Sensor Farbmodus:<br>IS_COLORMODE_BAYER |

| | | IS_COLORMODE_MONOCHROME |
|---|---|---|
| DWORD | nMaxWidth | Maximale Fenster Breite z.B. 1280 |
| DWORD | nMaxHeight | Maximale Fenster Höhe z.B. 1024 |
| BOOL | bMasterGain | Gemeinsamer Verstärker vorhanden ? - z.B. FALSE |
| BOOL | bRGain | Rot Verstärker vorhanden? - z.B. TRUE |
| BOOL | bGGain | Grün Verstärker vorhanden? - z.B. TRUE |
| BOOL | bBGain | Blau Verstärker vorhanden? - z.B. TRUE |
| BOOL | bGlobShutter | Global oder Rolling Shutter?<br>TRUE = GlobalShutter<br>FALSE = RollingShutter |
| Char | Reserved[16] | Reserviert |

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **pInfo** | Sensor Info Struktur |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.47. is_GetUsedBandwidth

**Syntax:**

INT is_GetUsedbandwidth (HIDS hf)

**Beschreibung:**

*is_GetUsedBandwidth()* gibt die momentan belegte Bandbreite durch den Pixelclock aller aktiven Kameras zurück.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |

**Rückgabewert:**

Summe des Pixeltaktes

## 4.48. is_GetVsyncCount

**Syntax:**

INT is_GetVsyncCount (HIDS hf, long* pIntr, long* pFrame)

**Beschreibung:**

*is_GetVsyncCount()* liest den VSYNC Zähler aus. Dieser wird bei jedem VSYNC um 1 erhöht.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **pIntr** | Aktueller VSYNC Zählerstand |
| **pFrame** | Aktueller Frame-SYNC Zählerstand |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.49. is_GetWhiteBalanceMultipliers

**Syntax:**

INT is_GetWhiteBalanceMultipliers (HIDS hf, double *pdblRed, double *pdblGreen, double *pdblBlue)

**Beschreibung:**

Die Funktion *is_GetWhiteBalanceMultipliers()* gibt die momentan verwendeten Faktoren des Weißabgleichs zurück, die entweder zuvor mit is_SetWhiteBalanceMultipliers() gesetzt, oder über den automatischen Weißabgleich berechnet wurden.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **pdblRed** | Faktor für Rotkanal. |
| **pdblGreen** | Faktor für Grünkanal. |
| **pdblBlue** | Faktor für Blaukanal. |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

 

## 4.50. is_HasVideoStarted

**Syntax:**

INT is_HasVideoStarted (HIDS hf, BOOL* pbo)

**Beschreibung:**

Mit *is_HasVideoStarted()* kann überprüft werden, ob die Digitalisierung des Bildes gestartet wurde. Diese Funktion ist im Zusammenhang mit *is_FreezeVideo()* mit dem Parameter *IS_DONT_WAIT* sinnvoll.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **pbo** | Enthält dann den Digitalisierstatus:<br>0 = Bildaufnahme noch nicht gestartet<br>1 = Bildaufnahme gestartet |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.51. is_HideDDOverlay

**Syntax:**

INT is_HideDDOverlay (HIDS hf)

**Beschreibung:**

*is_HideDDOverlay()* blendet das Overlay im DirectDraw BackBuffer-Modus aus. Es wird dann nur noch der Bildbuffer dargestellt, so dass die Bildwiederholrate höher ist als mit eingeblende-tem Overlay. Durch das Ausblenden des Overlays gehen die Overlaydaten nicht verloren.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.52. is_InitCamera

**Syntax:**

INT is_InitCamera (HIDS* phf, HWND hWnd)

**Beschreibung:**

*is_InitCamera()* öffnet den Treiber und stellt den Kontakt zur Hardware her. In dieser Funktion wird nach erfolgreicher Initialisierung das Handle auf die Kamera vergeben. Alle nachfolgenden Funktionen benötigen diesen Handle als ersten Parameter. Wenn kein DirectDraw zur Bildausgabe verwendet wird, kann *hWnd*=NULL übergeben werden.
Damit mehrere uEye Kameras parallel in einem System arbeiten können (Multikamera-Betrieb) muss bei der Initialisierung entschieden werden, welche Kamera initialisiert werden soll. Im EEPROM ist eine Kamera-ID festgelegt (siehe auch *4.24 is_GetCameraInfo* bzw. die Struktur *IDSINFO* mit dem Eintrag *Select*). Soll mit *is_InitCamera()* eine bestimmte Kamera angesprochen werden, so muss zuvor das Handle *hf* mit der gewünschten Kamera-ID initialisiert werden. Ist der Multikamera-Betrieb nicht erwünscht, muss das Handle *hf* vor dem Aufruf von *is_InitCamera()* mit 0 initialisiert werden. 0 bedeutet, dass die erste nicht benutzte Kamera verwendet wird.

Im Allgemeinen ist das uEye SDK *thread safe*. Die API-Funktionsaufrufe erfolgen alle in *critical sections*. Auf Grund interner *DirectDraw* Strukturen empfehlen wird dringend, die folgenden Funktionen nur aus einem *Thread* heraus aufzurufen, um ein unvorhersehbares Verhalten Ihrer Applikation zu vermeiden.

- *is_InitCamera()*
- *is_SetDisplayMode()*
- *is_ExitCamera()*

**Übergabeparameter:**

| | |
|---|---|
| **phf** | Zeiger auf den Handle der Kamera<br>Der Inhalt des Zeigers hat beim Aufruf der Funktion folgende Bedeutung:<br>0: verwende die erste freie Kamera<br>1-254: Öffne die Kamera mit dieser ID |
| **hWnd** | Handle auf das Fenster, in dem das Bild dargestellt werden soll<br>(kann NULL sein) |

**Rückgabewert:**

*IS_SUCCESS*, Fehlercode (siehe Header-Datei)

 

## 4.53. is_InitEvent

**Syntax:**

INT is_InitEvent (HIDS hf, HANDLE hEv, INT which)

**Beschreibung:**

Initialisiert den Event Handle durch Registrierung des im Parameter *which* angegebenen Event-Objekts im Kernel-Treiber.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **hEv** | Event-Handle von der C/C++-Funktion *CreateEvent()* |
| **which** | ID welches Event initialisiert werden soll: |
| IS_SET_EVENT_FRAME | Ein neues Bild steht zur Verfügung. |
| IS_SET_EVENT_SEQ | Die Sequenz wurde durchlaufen. |
| IS_SET_EVENT_STEAL | Ein dem Overlay entzogenes Bild steht zur Verfügung. |
| IS_SET_EVENT_TRANSFER_FAILED | Daten gingen während der Übertragung verloren, |
| IS_SET_EVENT_EXTTRIG | Ein Bild, dessen Aufnahme durch einen Trigger ausgelöst wurde, wurde komplett empfangen. Dies ist der frühest möglicher Zeitpunkt für eine neue Aufnahme. Das Bild muss noch das Postprocessing des Treibers durchlaufen und steht nach *IS_FRAME* zur Verarbeitung bereit. |
| IS_SET_EVENT_MEMORY_MODE_FINISH | Die Bildaufnahme in das optionale Memorymodul wurde beendet. |
| IS_SET_EVENT_REMOVE | Eine mit *is_InitCamera()* geöffnete Kamera wurde entfernt. |
| IS_SET_EVENT_DEVICE_RECONNECTED | Eine mit *is_InitCamera()* geöffnete und danach entfernte Kamera wurde wieder eingesteckt. |
| IS_SET_EVENT_NEW_DEVICE | Eine Kamera wurde neu angeschlossen.<br>Unabhängig vom Device Handle (*hf* wird ignoriert). |
| IS_SET_EVENT_REMOVAL | Eine Kamera wurde entfernt. Unabhängig vom Device Handle (*hf* wird ignoriert). |
| IS_SET_EVENT_WB_FINISHED | Der automatische Weißabgleich hat seine Regelung beendet. |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

**Beispiel:**

Frame event aktivieren, Bildaufnahme starten und auf Event warten:

```
HANDLE hEvent = CreateEvent(NULL, TRUE, FALSE, "");
if (hEvent != NULL) {
  is_InitEvent(hf, hEvent, IS_SET_EVENT_FRAME);
  is_EnableEvent(hf, IS_SET_EVENT_FRAME);
  is_FreezeVideo(hf, IS_DON_T_WAIT);
  if (WaitForSingleObject(hEvent, 1000) == WAIT_OBJECT_0) {
    // Bild Erfolgreich aufgenommen }
  is_DisableEvent(hf, IS_SET_EVENT_FRAME);
  is_ExitEvent(hf, IS_SET_EVENT_FRAME); }
```

 

## 4.54. is_InquireImageMem

**Syntax:**

INT is_InquireImageMem (HIDS hf, char* pcMem, int nID, int* pnX, int* pnY, int* pnBits, int* pnPitch);

**Beschreibung:**

*is_InquireImageMem()* liest die Eigenschaften eines allokierten Bildspeichers aus.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **pMem** | Zeiger auf den Anfang des Bildspeichers von *is_AllocImageMem()* |
| **NID** | ID des Bildspeichers von *is_AllocImageMem()* |
| **pnX** | Enthält die Breite, mit der der Bildspeicher angelegt wurde (kann NULL sein) |
| **pnY** | Enthält die Höhe, mit der der Bildspeicher angelegt wurde (kann NULL sein) |
| **pnBits** | Enthält die Bitbreite, mit der der Bildspeicher angelegt wurde (kann NULL sein) |
| **pnPitch** | Enthält den Zeileninkrement des Bildspeichers (kann NULL sein) |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.55. is_IsMemoryBoardConnected

**Syntax:**

INT is_IsMemoryBoardConnected (HIDS hf, BOOL* pConnected)

**Beschreibung:**

Mit der Funktion *is_IsMemoryBoardConnected()* kann abgefragt werden ob das optionale Memoryboard vorhanden ist.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **pConnected** | TRUE = Memoryboard ist angeschlossen<br>FALSE = Kein Memoryboard vorhanden |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.56. is_IsVideoFinish

**Syntax:**

INT is_IsVideoFinish (HIDS hf, BOOL* pbo)

**Beschreibung:**

Mit *is_IsVideoFinish()* kann überprüft werden, ob ein Bild vollständig in den Bildspeicher erfasst wurde. Diese Funktion ist im Zusammenhang mit *is_FreezeVideo()* mit dem Parameter *IS_DONT_WAIT* sinnvoll.
Wenn vor dem Aufruf von *is_IsVideoFinish()* *pbo = IS_TRANSFER_FAILED gesetzt wird, kann zusätzlich geprüft werden, ob ein Übertragungsfehler oder ein Fehler im Pixelpfad aufgetreten ist.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **pbo** | *pbo != IS_TRANSFER_FAILED vor dem Funktionsaufruf<br>pbo enthält den Digitalisierstatus:<br>IS_VIDEO_NOT_FINISH = Bild noch nicht fertig digitalisiert<br>IS_VIDEO_FINISH = Bild ist fertig digitalisiert<br><br>*pbo == IS_TRANSFER_FAILED vor dem Funktionsaufruf<br>pbo enthält den Digitalisierstatus:<br>IS_VIDEO_NOT_FINISH = Bild noch nicht fertig digitalisiert<br>IS_VIDEO_FINISH = Bild ist fertig digitalisiert<br>IS_TRANSFER_FAILED = Übertragungsfehler oder Problem beim Konvertieren (z.B. Zielspeicher ungültig) |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.57. is_LoadBadPixelCorrectionTable

**Syntax:**

INT is_LoadBadPixelCorrectionTable (HIDS hf, char *File)

**Beschreibung:**

*is_LoadBadPixelCorrectionTable()* lädt eine zuvor mit der Funktion *SaveBadPixelCorrection-Table()* gespeicherte Tabelle. *File* gibt die Datei an, in der die Koordinaten gespeichert wurden, bei Übergabe eines NULL Zeigers, wird ein Dialog zur Auswahl der Datei angezeigt.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **File** | Zeiger auf Datei mit gespeicherten Koordinaten. Es kann sowohl der absolute als auch der relative Pfad übergeben werden. |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.58. is_LoadImage

**Syntax:**

INT is_LoadImage (HIDS hf, char* File)

**Beschreibung:**

Lädt ein Bild aus einer Datei. Das Bild muss im BMP-Format vorliegen. Das Bild wird in den aktiven Bildspeicher geladen (siehe auch *4.37 is_GetImageMem* und *4.110 is_SetImageMem*).

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **File** | Zeiger auf Dateiname. Es kann sowohl der absolute als auch der relative Pfad übergeben werden. |

**Rückgabewert:**

| | |
|---|---|
| IS_SUCCESS | Bild wurde fehlerfrei geladen. |
| IS_FILE_READ_INVALID_BMP_SIZE | Das zu ladende Bild ist größer als der aktive Bildspeicher. |
| IS_FILE_READ_INVALID_BMP_ID | Die zu ladende Datei besitzt kein gültiges Bitmap-Format. |
| IS_FILE_READ_OPEN_ERROR | Die Datei konnte nicht geöffnet werden. |

# 4.59. is_LoadImageMem

**Syntax:**

INT is_LoadImageMem (HIDS hf, char* File, char** ppcImgMem, int* pid)

**Beschreibung:**

*is_LoadImage()* lädt ein Bild aus einer Datei, welches im BMP-Format vorhanden sein muss. Das Bild wird mit den Eigenschaften Farbformat und -tiefe in einen neu allokierten Bildspeicher geladen.
Mit der *Funktion is_FreeImageMem()* (siehe *4.18 is_FreeImageMem*) wird der Bildspeicher wieder freigegeben.

**Übergabeparameter:**

| | |
|---|---|
| **Hf** | Handle auf Kamera |
| **File** | Dateiname. Es kann sowohl der absolute als auch der relative Pfad übergeben werden. |
| **ppcImgMem** | Zeiger auf eine Variable mit der Startadresse |
| **Pid** | Zeiger auf eine Variable mit der Speicher ID |

**Rückgabewert:**

IS_SUCCESS (Das Bild wurde fehlerfrei geladen)
IS_FILE_READ_INVALID_BMP_ID (Das Bitmapformat des zu ladenden Bildes ist nicht gültig)
IS_FILE_READ_OPEN_ERROR (Die Datei kann nicht geöffnet werden)

## 4.60. is_LoadParameters

**Syntax:**

INT is_LoadParameters (HIDS hf, char* pFilename)

**Beschreibung:**

*is_LoadParameters()* lädt die Parameter einer Kamera, die zuvor mit *is_SaveParameters()* als ini-Datei gespeichert wurden. Wird für *pFilename* der Wert NULL übergeben, wird der *Datei öffnen* Dialog von Windows angezeigt.

> Es können nur kameraspezifische ini-Dateien geladen werden.
> Beim Laden einer ini-Datei ist zu berücksichtigen, dass bereits allokierter Speicher den Parametern des ini-Files hinsichtlich Bildgröße (AOI) und Farbtiefe übereinstimmt. Wenn das nicht der Fall ist, führt dies zu Darstellungsfehlern.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **pFilename** | Zeiger auf Dateiname. Es kann sowohl der absolute als auch der relative Pfad übergeben werden. |

**Rückgabewert:**

IS_SUCCESS, IS_NO_SUCCESS

**Beispiel ini-Datei:**

```
[Sensor]
Sensor=UI122x-M
[Image size]
Start X=0
Start Y=0
Start X absolute=0
Start Y absolute=0
Width=752
Height=480
Binning=0
Subsampling=0
[Timing]
Pixelclock=20
Framerate=30.003810
Exposure=33.329100
[Parameters]
Colormode=6
Brightness=100
Contrast=215
Gamma=1.000000
Blacklevel Mode=1
Blacklevel Offset=0
```

```
[Gain]
Master=0
Red=0
Green=0
Blue=0
[Processing]
EdgeEnhancement=0
RopEffect=0
Whitebalance=0
Whitebalance Red=1.000000
Whitebalance Green=1.000000
Whitebalance Blue=1.000000
Color correction=0
[Auto features]
Auto Framerate control=0
Brightness exposure control=0
Brightness gain control=0
Brightness reference=128
Brightness speed=50
Brightness max gain=-1.000000
Brightness max exposure=1.000000
Brightness Aoi Left=0
Brightness Aoi Top=0
Brightness Aoi Width=1280
Brightness Aoi Height=1024
Auto WB control=0
Auto WB offsetR=0
Auto WB offsetB=0
Auto WB gainMin=0
Auto WB gainMax=100
Auto WB speed=50
Auto WB Aoi Left=0
Auto WB Aoi Top=0
Auto WB Aoi Width=1280
Auto WB Aoi Height=1024
Auto WB Once=0
```

## 4.61. is_LockDDMem

**Syntax:**

INT is_LockDDMem (HIDS hf, void** ppMem, INT* pPitch)

**Beschreibung:**

*is_LockDDMem()* gibt den Zugriff auf den Bildspeicher in den DirectDraw Modi frei und liefert den Adresszeiger auf den Anfang des Bildspeichers zurück. Der Bildspeicher befindet sich in den meisten Fällen auf der VGA Karte. Mit dem Zeiger kann direkt auf den Bildspeicher zugegriffen werden. Der Zugriff muss sobald als möglich mit der Funktion *is_UnlockDDMem()* aufgehoben werden.
Die Digitalisierung in den Speicherbereich wird durch einen Aufruf von *is_LockDDMem()* nicht unterbrochen!
Im DirectDraw Backbuffer Modus erfolgt innerhalb eines *LockDDMem - UnlockDDMem* Blocks erfolgt KEIN Update des Backbuffers auf dem Bildschirm!

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **ppMem** | Zeiger auf Variable, die den Adresszeiger aufnimmt |
| **pPitch** | Zeiger auf Variable, die den Pitch-Wert aufnimmt |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.62. is_LockDDOverlayMem

**Syntax:**

INT is_LockDDOverlayMem(HIDS hf, void** ppMem, INT* pPitch)

**Beschreibung:**

*is_LockDDOverlayMem()* gibt im DirectDraw Backbuffer-Modus den Zugriff auf den Overlay-speicher frei und liefert den Adresszeiger auf den Anfang des Overlay-Buffers zurück. Damit kann direkt in den Overlay-Buffer geschrieben werden, ohne die Windows-GDI Funktionen verwenden zu müssen. pPitch liefert den Zeilenoffset in Bytes vom Anfang einer Zeile zum Anfang der folgenden Zeile zurück. Der Zugriff muss sobald als möglich mit der Funktion *is_UnlockDDOverlayMem()* wieder aufgehoben werden. Innerhalb eines *LockDDOverlayMem - UnlockDDOverlayMem* Blocks erfolgt KEIN Update des Overlaybuffers auf dem Bildschirm.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **ppMem** | Zeiger auf Variable, die den Adresszeiger aufnimmt |
| **pPitch** | Zeiger auf Variable, die den Pitch-Wert aufnimmt |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.63. is_LockSeqBuf

**Syntax:**

INT is_LockSeqBuf (HIDS hf, INT nNum, char* pcMem)

**Beschreibung:**

Mit *is_LockSeqBuf()* kann ein Bildspeicher vor dem Überschreiben mit neuen Bilddaten gesperrt werden, d.h. dieser Bildspeicher wird in der Sequenzliste der zu verwendenden Bildspeicher ausgeklammert. Damit kann verhindert werden, dass Bilddaten, die noch zur weiteren Verarbeitung benötigt werden, durch neue Daten überschrieben werden. Auf den Bildspeicher kann weiterhin voll zugegriffen werden.
Es kann immer nur ein Bildspeicher gleichzeitig gesperrt sein. Zum Freigeben dient die Funktion *is_UnlockSeqBuf()*.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **nNum** | Nummer des Bildspeichers, der geschützt werden soll (1...max) |
| **pcMem** | Startadresse des Bildspeichers, der geschützt werden soll |

nNum bezeichnet die Position in der Sequenzliste und nicht die mit *is_AllocImageMem()* vergebene Speicher-ID.

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.64. is_MemoryFreezeVideo

**Syntax:**

INT is_MemoryFreezeVideo (HIDS hf, INT nMemID, INT Wait)

**Beschreibung:**

Mit der Funktion *is_MemoryFreezeVideo()* kann mit einem einzigen Befehl ein Einzelbild über das Memoryboard aufgenommen werden. Dabei wird zuerst ein Bild in den Kameraspeicher übertragen und anschließend in den angegeben Bildspeicher eingelesen.
Der Vorteil dieser Funktion besteht darin, dass ein aufgenommenes Bild auch bei starker Rechnerbelastung ohne Transferfehler sicher übertragen werden kann.
Erfolgt nach dem Aufruf der Funktion mit dem Parameter *IS_DONT_WAIT* eine Veränderung der Kameraeinstellungen (Exposure, Pclk, AOI, ...) wird die Aufnahme abgebrochen und die neuen Kameraeinstellungen werden übernommen.

Bei Aufruf der Funktion *is_MemoryFreezeVideo()* wird automatisch der Memorymode aktiviert. Um anschließend wieder eine Bilderfassung ohne Memorymode zu ermöglichen, muss der Memorymode mit der Funktion *is_SetMemoryMode(IS_MEMORY_MODE_DISABLE, 0)* (siehe *4.116 is_SetMemoryMode*) wieder abgeschaltet werden.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **nMemID** | ID des Speichers in den das Bild übertragen wird, bei 0 wird der momentan aktive Speicher verwendet. |
| **Wait** | |
| IS_WAIT | Funktion wartet bis Bild aufgenommen ist |
| IS_DONT_WAIT | Funktion kehrt sofort zurück |
| 10 <= Wait < 21474836 | Wartezeit in 10 ms-Schritten. Maximal kann 214748,36Sekunden gewartet werden.<br>Für 1 < Wait < 10 wird Wait = 10 gesetzt.<br>(Bsp.: Wait = 100 ⇒ 1 sec warten) |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.65. is_PrepareStealVideo

**Syntax:**

INT is_PrepareStealVideo (HIDS hf, int Mode, ULONG StealColorMode)

**Beschreibung:**

Setzt den Steal-Modus. Es gibt zwei unterschiedliche Steal-Modi:

- normal steal
  Mit dieser Option wird ein einzelnes Bildes aus einem DirectDraw Livestream in den aktuell aktiven Benutzerspeicher umgeleitet.
- copy steal
  Hier wird die Abbildung mit DirectDraw angezeigt und in den aktuell aktiven Bildspeicher kopiert.

**Übergabeparameter:**

| | |
|---|---|
| **Hf** | Handle auf Kamera |
| **Mode** | |
| IS_SET_STEAL_NORMAL | Normal Modus |
| IS_SET_STEAL_COPY | Copy Modus |
| **StealColorMode** | reserviert |

**Rückgabewert:**

IS_SUCCESS, IS_NO_SUCCESS

## 4.66. is_ReadEEPROM

**Syntax:**

INT is_ReadEEPROM (HIDS hf, INT Adr, char* pcString, INT Count)

**Beschreibung:**

In der Kamera befindet sich ein EEPROM als kleiner Speicher. Neben den fest von IDS hinterlegten Informationen können 64 Bytes eigene Daten in das EEPROM geschrieben werden. Mit der Funktion *is_ReadEEPROM()* kann der Inhalt dieses 64-Byte Blockes gelesen werden. Siehe auch *4.134 is_WriteEEPROM*.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **Adr** | Anfangsadresse, ab der Daten gelesen werden soll. Wertebereich 0..63 |
| **pcString** | Buffer für die zu lesenden Daten (min. Größe = Count) |
| **Count** | Anzahl zu lesender Zeichen |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

**Beispiel:**

```
char buffer[64];
is_ReadEEPROM(m_hCam, 0x00, buffer, 64);
```

## 4.67. is_ReadI2C (nur uEyeLE)

**Syntax:**

INT is_ReadI2C (HIDS hf, INT nDeviceAddr, INT nRegisterAddr, BYTE* pbData, INT nLen)

**Beschreibung:**

Mit *is_ReadI2C()* können Daten über den $I^2C$-Bus gelesen werden. $I^2C$-Bustakt beträgt 100 kHz.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **nDeviceAdr** | Slave Adresse |
| **nRegisterAddr** | Register Adresse (nur 8 Bit-Adressen gültig). |
| **data** | Zu lesende Daten |
| **length** | Datenlänge |

Ungültige Adresssen für *nRegisterAddr*:
0x48, 0x4C, 0x51, 0x52, 0x55, 0x5C, 0x5D, 0x69 (diese werden intern verwendet).

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS* or *IS_INVALID_I2C_DEVICE_ADDRESS*

## 4.68. is_ReleaseDC

**Syntax:**

INT is_ReleaseDC (HIDS hf, HDC hDC)

**Beschreibung:**

*is_ReleaseDC()* gibt im DirectDraw BackBuffer-Modus das Device-Context-Handle des Overlay Buffers im DirectDraw BackBuffer Modus frei. Nach Freigabe des Handles erfolgt ein Update des Overlay Buffers auf dem Bildschirm (wenn mit *is_ShowDDOverlay()* das Overlay eingeblendet ist)

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **hDC** | Device-Context-Handle von *is_GetDC()* |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.69. is_RenderBitmap

**Syntax:**

INT is_RenderBitmap (HIDS hf, INT nMemID, HWND hwnd, INT nMode)

**Beschreibung:**

Mit *is_RenderBitmap()* kann ein Bild aus einem Bildspeicher in einem angegebenen Fenster ausgegeben werden. Zur Darstellung werden Win32 Bitmapfunktionen verwendet. Die Darstellung erfolgt generell in dem Format, das beim Allokieren des Bildspeichers festgelegt wurde. In der Funktion *is_AllocImageMem()* legt der Parameter *bitspixel* die Farbtiefe und die Darstellungsart fest. RGB16 und RGB 15 benötigen denselben Speicherplatz, können aber anhand des Parameters *bitspixel* differenziert werden.

> Das Farbformat UYVY ist nur im Overlaymode sinnvoll darzustellen. Dieser ist mit den in *is_RenderBitmap()* verwendeten Funktionen der Windows-API aber nicht realisierbar.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **nMemID** | ID des Bildspeichers der angezeigt werden soll |
| **hwnd** | Fenster Handle des Ausgabefensters |
| **nMode** | |
| IS_RENDER_NORMAL | Bild normal ausgeben. Es wird 1:1 dargestellt, so wie das Bild im Bildspeicher vorliegt. |
| IS_RENDER_FIT_TO_WINDOW | Bild in das Ausgabefenster einpassen. |
| IS_RENDER_ DOWNSCALE_1_2 | Bild mit 50% der Originalgröße Ausgeben. |
| Optionen, die mit den o.g. logisch ODER verknüpft werden können: | |
| IS_RENDER_MIRROR_UPDOWN | Gibt das Bild an der horizontalen Achse gespiegelt aus (upside down) |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

**Beispiel:**

Bild in Fenster einpassen und upside down darstellen:

```
is_RenderBitmap (hf, nMemID, hwnd, IS_RENDER_FIT_TO_WINDOW |
IS_RENDER_MIRROR_UPDOWN);
```

## 4.70. is_ResetMemory

**Syntax:**

INT is_ResetMemory (HIDS hf, INT reserved)

**Beschreibung:**

Gespeicherte Bilddaten bleiben im Speicher des Memoryboards erhalten, solange die Kamera mit Strom versorgt wird. Dadurch ist der Zugriff auf die Daten solange möglich, bis der Speicher des Memoryboards durch den Aufruf von *is_ResetMemory()* gelöscht wird.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **reserved** | Wird aktuell nicht verwendet. |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.71. is_ResetToDefault

**Syntax:**

INT is_ResetToDefault (HIDS hf)

**Beschreibung:**

*is_ResetToDefault()* setzt alle Parameter auf die kameraspezifischen Standardwerte zurück.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.72. is_SaveBadPixelCorrectionTable

**Syntax:**

INT is_SaveBadPixelCorrectionTable (HIDS hf, char *File)

**Beschreibung:**

*is_SaveBadPixelCorrectionTable()* speichert die aktuelle, benutzerdefinierte HotpixelListe in die mit dem Parameter *File* angegebene Datei.
Bei Übergabe des Wertes NULL für den Parameter *File* wird ein Dialog zur Dateiauswahl angezeigt.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **File** | Zeiger auf die Datei, in der die Daten gespeichert werden. Es kann sowohl der absolute als auch der relative Pfad übergeben werden. |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.73. is_SaveImage

**Syntax:**

INT is_SaveImage (HIDS hf, char* File)

**Beschreibung:**

Speichert ein Bild im Bitmap Format (*.BMP) in eine Datei. Die Bilder werden aus dem aktiven Bildspeicher ausgelesen (siehe auch *4.37 is_GetImageMem*). In den DirectDraw Modi werden die Bilddaten direkt aus dem entsprechenden DirectDraw Buffer gelesen. Das Auslesen des Grafikartenspeichers kann, je nach Bildgröße, mehrere 100msec dauern. Im DirectDraw Primary Surface Modus ist zu beachten, dass das Bild so gespeichert wird wie es aktuell auf dem Bildschirm dargestellt ist. Es wird nur mit maximal der Größe des Bild-Ausgabefensters gespeichert, unabhängig davon welche Bildgröße mit *is_SetImageSize()* eingestellt ist. Falls andere Fenster das Bild-Ausgabefenster ganz oder teilweise überlagern, so wird der überlagernde Inhalt mit gespeichert!
Das Bitmap wird mit der Farbtiefe (8, 15, 16, 24 oder 32 Bit) gespeichert, wie der Bildspeicher allokiert wurde bzw. wie der aktuelle Farbmodus bei DirectDraw Ausgabemodi eingestellt ist. Einige Bildbearbeitungsprogramme unterstützen keine 15 Bit, 16 Bit oder 32 Bit Bitmaps und können somit in diesen Modi gespeicherte Bilder nicht einlesen!
Der Dateiname kann sowohl absolute als auch relative Pfadangaben enthalten. Mit *is_LoadImage()* werden BMP-Bilder gelesen.
Overlay-Daten werden nicht gespeichert!

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **File** | Name der BMP-Bilddatei |
| | NULL -> "Speichern unter"-Dialogbox wird geöffnet. |
| | Es kann sowohl der absolute als auch der relative Pfad übergeben werden. |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.74. is_SaveImageEX

**Syntax:**

INT is_SaveImageEx (HIDS hf, char* File, INT fileFormat, INT Param)

**Beschreibung:**

Speichert ein Bild als Bitmap oder JPG in eine Datei (siehe auch *4.73 is_SaveImage*). Die Bilder werden aus dem aktiven Bildspeicher ausgelesen. Das Bitmap wird mit der Farbtiefe (8, 15, 16, 24 oder 32 Bit) gespeichert, wie der Bildspeicher allokiert wurde bzw. wie der aktuelle Farbmodus bei DirectDraw Ausgabemodi eingestellt ist. Einige Bildbearbeitungsprogramme unterstützen keine 15 Bit, 16 Bit oder 32 Bit Bitmaps und können somit in diesen Modi gespeicherte Bilder nicht einlesen. Der Dateiname kann sowohl absolute als auch relative Pfadangaben enthalten.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **File** | Name der BMP-Bilddatei<br>NULL -> "Speichern unter"-Dialogbox wird geöffnet.<br>Es kann sowohl der absolute als auch der relative Pfad übergeben werden. |
| **fileFormat** | bestimmt das Ausgabeformat der Datei |
| IS_IMG_BMP | Bitmap |
| IS_IMG_JPG | JPEG |
| **Param** | Wenn JPEG als Dateiformat gewählt wird, kann mit *Param* die Qualität zwischen 1 und 100 eingestellt werden. Wenn *Param* 0 ist, wird die voreingestellt Qualität (75) verwendet. |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*
*IS_INVALID_PARAMETER* (üngültites Dateiformat oder ungültige JPEG Qualität)

## 4.75. is_SaveImageMem

**Syntax:**

INT is_SaveImageMem (HIDS hf, char* File, char* pcMem, int nID)

**Beschreibung:**

Speichert ein Bild im Bitmap Format (*.BMP) in eine Datei. Die Bilder werden aus dem angegebenen Bildspeicher ausgelesen
Das Bitmap wird mit der Farbtiefe (8, 15, 16, 24 oder 32 Bit) gespeichert, wie der Bildspeicher allokiert wurde. Einige Bildbearbeitungsprogramme unterstützen keine 15 Bit, 16 Bit oder 32 Bit Bitmaps und können somit in diesen Modi gespeicherte Bilder nicht einlesen!
Overlay-Daten werden nicht gespeichert!

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **File** | Name der BMP-Bilddatei |
| | NULL -> "Speichern unter"-Dialogbox wird geöffnet. |
| | Es kann sowohl der absolute als auch der relative Pfad übergeben werden. |
| **pcMem** | Zeiger auf Bildspeicher |
| **nID** | ID des Bildspeichers (USE_ACTUAL_IMAGE_SIZE kann mit nID logisch ODER verknüpft werden, um das Bild mit der aktuell eingestellten Bildhöhe zu speichern) |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

# 4.76. is_SaveImageMemEx

**Syntax:**

INT is_SaveImageMemEx (HIDS hf, char* File, char* pcMem, int nID,INT fileFormat, INT Param).

**Beschreibung:**

*is_SaveImageMemEx()* speichert ein Bild im Bitmap- oder JPEG-Format. Die Bilder werden vom Bildspeicher gelesen. Das Bitmap wird mit einer Farbtiefe von 8, 15, 16, 24 oder 32 Bit gespeichert, entsprechend dem allokierten Bildspeicher. Einige Bildverarbeitungsprogramme unterstützen nicht alle Farbformate (15, 16, 32 Bit) und können diese Bilder nicht laden. Die JPEG-Datei wird immer mit einer Farbtiefe von 8 oder 24 Bit gespeichert.

**Übergabeparameter:**

| | |
|---|---|
| **Hf** | Handle auf Kamera |
| **File** | Name der Bilddatei<br>NULL -> "Speichern unter" Dialog wird geöffnet.<br>Es kann sowohl der absolute als auch der relative Pfad übergeben werden. |
| **pcMem** | Zeiger auf Bildspeicher |
| **nID** | ID des Bildspeichers |
| **fileFormat** | bestimmt das Ausgabeformat der Datei |
| IS_IMG_BMP | Bitmap |
| IS_IMG_JPG | JPEG |
| **Param** | Wenn JPEG als Dateiformat gewählt wird, kann mit *Param* die Qualität der Kompression zwischen 1 und 100 eingestellt werden. Wenn *Param* 0 ist, wird die voreingestellte Qualität (75) verwendet. |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*
*IS_INVALID_PARAMETER* (ungültiges Dateiformat oder ungültige JPEG Qualität)

## 4.77. is_SaveParameters

**Syntax:**

INT is_SaveParameters (HIDS hf, char* pFilename)

**Beschreibung:**

*is_SaveParameters()* speichert die Parameter einer Kamera, in eine ini-Datei, die zu einem späteren Zeitpunkt mit *is_LoadParameters()* geladen werden kann. Wird für *pFilename* der Wert NULL übergeben, wird der *Datei speichern* Dialog von Windows angezeigt.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **pFilename** | Zeiger auf Dateiname. Es kann sowohl der absolute als auch der relative Pfad übergeben werden. |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

**Beispiel ini-Datei:**

Siehe *4.60* *is_LoadParameters*

## 4.78. is_SetAllocatedImageMem

**Syntax:**

INT is_SetAllocatedImageMem (HIDS hf, INT width, INT height, INT bitspixel, char* pcImgMem, int* pid)

**Beschreibung:**

*is_SetAllocatedImageMem()* erklärt einen vom Benutzer allokierten Speicher zu einem Speicher, in den digitalisiert werden kann. Der Speicher muss groß genug allokiert worden und muss unbedingt global gelocked sein! Ein mit *is_SetAllocatedImageMem()* angegebener Speicher kann in eine Sequenz eingebaut werden. Der Speicher muss mit *is_FreeImageMem()* aus der treiberinternen Verwaltung genommen werden, bevor er tatsächlich wieder freigegeben wird. Der allokierte Speicher darf nicht reallokiert werden.
Folgende Reihenfolge ist zu beachten:

- Speicher allokieren: HANDLE hgMem = GlobalAlloc (size);
- Speicher locken: char* pcMem = (char*) GlobalLock (hgMem);

Die Adresse des Speichers wird an den uEye-Treiber übergeben. Dies geschieht mit der Funktion *is_SetAllocatedImageMem()*. Zusätzlich muss wie bei *is_AllocImageMem()* die Größe des Bildes angegeben werden. Eine Speicher-ID wird zurückgegeben. Diese wird für andere Funktionen evtl. benötigt.
*is_SetAllocatedImageMem (hf, width, height, bitspixel, pcMem, &ID)*;

size >= (width * height * bitspixel / 8)

Der Bildspeicher kann nun wie üblich verwendet werden. Sequenzen sind ebenso möglich. Der Speicherbereich muss mit der Funktion *is_FreeImageMem (hf, pcMem, ID)* wieder aus der Treiber-Verwaltung genommen werden. Hierbei wird der Speicher jedoch NICHT freigegeben. Der Benutzer muss dafür sorgen, dass der Speicher freigegeben wird:

- GlobalUnlock (*hgMem*);
- GlobalFree (*hgMem*);

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **width** | Breite des Bildes |
| **height** | Höhe des Bildes |
| **bitspixel** | Farbtiefe des Bildes (Bits pro Pixel) |
| **pcImgMem** | Zeiger auf den Speicheranfang des allokierten Speichers |
| **pid** | Enthält dann die ID für diesen Speicher |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

 

## 4.79. is_SetAOI

**Syntax:**

INT is_SetAOI (HIDS hf, INT type, INT *pXPos, INT *pYPos, INT *pWidth, INT *pHeight)

**Beschreibung:**

Mit *is_SetAOI()* kann die Größe und Position eines AOI über einen Befehlsaufruf gesetzt werden. Mögliche AOI sind:

- Image AOI
- Auto Brightness AOI
- Auto Whitebalance AOI
  Die Auto-Bildbereiche werden für die entsprechende Autofunktionalität verwendet. Als Defaultwert ist das Fenster immer maximal, also immer so groß wie das aktuelle Image AOI.

Nach Änderungen der Bildgeometrie, z.B. durch Neusetzen eines Image AOI werden die Auto Feature AOI immer auf den Wert des zuvor eingestellten Image AOI zurückgesetzt. Das bedeuted, dass die Auto Feature AOI eventuell neu gesetzt werden müssen.

Neben Image AOI sind Binning und Subsampling weitere, die Bildgeometrie beeinflussende Funktionen.

Auto Whitebalance AOI ist im DirectDraw Modus und im UYVY-Modus nicht verfügbar.

Mit den Funktionen *is_SetImagePos()* (siehe *4.111 is_SetImagePos*) und *is_SetImageSize()* (siehe *4.112 is_SetImageSize*) können Informationen über die Größe und Position des AOI ausgelesen werden.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **type** | |
| IS_SET_IMAGE_AOI | Image AOI setzen |
| IS_GET_IMAGE_AOI | Gibt das aktuelle Image AOI zurück |
| IS_SET_AUTO_BRIGHT_AOI | Mittelwert AOI für Auto Gain und Auto Shutter setzen |
| IS_GET_AUTO_BRIGHT_AOI | Gibt das aktuelle Mittelwert AOI zurück |
| IS_SET_AUTO_WB_AOI | AOI für Auto Whitebalance setzen |
| IS_GET_AUTO_WB_AOI | Gibt das aktuelle Auto Whitebalance AOI zurück |
| **pXPos** | Horizontale Position des AOI |
| **pYPos** | Vertikale Position des AOI |
| **pWidth** | Breite des AOI |
| **pHeight** | Höhe des AOI |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

**Beispiele:**

Auto Brightness AOI setzen:

```
Int nRet = SetAOI (m_hCam,              // Kamera Handle
  IS_SET_AUTO_BRIGHT_AOI,               // AOI Typ + Kommando
  &xpos,
  &ypos,
  &width,
  &height);
```

## 4.80. is_SetAutoParameter

**Syntax:**

INT is_SetAutoParameter (HIDS hf, INT param, double* pval1, double* pval2)

**Beschreibung:**

*is_SetAutoParameter()* steuert die Gain, Shutter, Framerate und Whitebalance Autofunktionen. Das Ziel der Auto Funktionen ist es, das Kamerabild in seiner mittleren Helligkeit und Farbwiedergabe auf den vorgegebenen Sollwert zu regeln und dabei die Bildwiederholrate (Framerate) auf einem maximalen Wert zu halten. Bei der Regelung der mittleren Helligkeit wird die Exposure Regelung bevorzugt. D.h. es wird erst die maximal mögliche Belichtungszeit eingestellt, bevor der Gain geregelt wird. AutoGain regelt den MasterGain der Kamera im Bereich 0-100%. Auto Exposure benutzt den jeweils aktuellen Exposure Bereich, der sich aus Pixelclock und Bildwiederholrate ergibt. Die Regelbereichsgrenzen können für Exposure und Gain getrennt eingestellt werden.
Die Framerate kann bei aktivierter Shutterregelung weiterhin manuell oder automatisch verändert werden, um den Regelbereich für die Belichtung dynamisch zu halten. Die automatische Frameratenregelung hat das Ziel, die Framerate auf einen optimalen Wert einzustellen. Dadurch steht der Shutterregelung in allen Situationen der erforderliche Regelbereich bei möglichst hoher Framerate zur Verfügung.
Bei der Regelung des Weißabgleichs werden die RGB-Gain Einstellungen der Kamera im Bereich 0-100% solange verändert, bis der rote, bzw. der blaue Kanal die mittlere Helligkeit des grünen Kanals erreichen. Um eine gewünschte Farbwiedergabe zu erhalten, können die Sollwerte für den roten und den blauen Kanal durch einen Offset relativ zum grünen Kanal eingestellt werden.
Die Geschwindigkeiten der Autofunktionen können in einem Bereich von 0 – 100% eingestellt werden. Dadurch wird die Dämpfung oder Trägheit der Regelung beeinflusst. Hohe Geschwindigkeit (100%) resultiert in einer kleinen Dämpfung für eine schnell reagierende Regelung und umgekehrt. Hierbei verwenden die Regelung für die mittlere Helligkeit und die Regelung für die Farbwiedergabe getrennte Geschwindigkeiten.
Durch das Setzen des Parameters *IS_SET_AUTO_WB_ONCE* kann die Regelung des Weißabgleichs automatisch beendet werden, sobald die Zielvorgabe erreicht wurde. Das Ende der Regelung wird dem System über ein Event/Message mitgeteilt (siehe auch *4.53 is_InitEvent)*. Alternativ bleibt die Regelung weiterhin aktiv und reagiert auf Abweichungen von der Zielvorgabe.

> Bei eingeschaltetem AutoShutter ist die Einstellung des Pixelclock über die Funktion *is_SetPixelClock()* deaktiviert.
>
> AutoFramerate ist nur bei eingeschalteter AutoShutter Regelung möglich, da die AutoFramerate den Shutterbereich dynamisch einstellt und AutoGain niemals aktiviert würde. Deshalb sind diese beiden Features gegenseitig verriegelt.
>
> AutoGain ist nur für Kameras mit MasterGain Einstellung möglich!
>
> Auto Whitebalance ist nur für Kameras mit Hardware RGB-Gain Einstellung möglich. Bei eingeschaltetem Auto Whitebalance ist das Einschalten der Software Whitebalance über die Funktion *is_SetWhiteBalance()* deaktiviert. Diese wird auch abgeschaltet, wenn sie beim Einschalten der Auto Whitebalance aktiv ist.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **param** | |
| IS_SET_ENABLE_AUTO_GAIN | Aktiviert/deaktiviert die Auto Gain Funktion |
| IS_GET_ENABLE_AUTO_GAIN | Gibt die aktuelle Auto Gain Einstellung zurück |
| IS_SET_ENABLE_AUTO_SHUTTER | Aktiviert/deaktiviert die Auto Shutter Funktion |
| IS_GET_ENABLE_AUTO_SHUTTER | Gibt die aktuelle Auto Shutter Einstellung zurück |
| IS_SET_ENABLE_AUTO_WHITEBALANCE | Aktiviert/deaktiviert die Auto Whitebalance Funktion |
| IS_GET_ENABLE_AUTO_WHITEBALANCE | Gibt die aktuelle Auto Whitebalance Einstellung zurück |
| IS_SET_ENABLE_AUTO_FRAMERATE | Aktiviert/deaktiviert die Auto Framerate Funktion |
| IS_GET_ENABLE_AUTO_FRAMERATE | Gibt die aktuelle Auto Framerate Einstellung zurück |
| IS_SET_AUTO_REFERENCE | Sollwert für AutoGain/AutoShutter setzen. Entspricht einem Grauwert von 0-255. |
| IS_GET_AUTO_REFERENCE | Gibt den Sollwert für AutoGain/AutoShutter zurück |
| IS_SET_AUTO_GAIN_MAX | Obere Regelgrenze für AutoGain setzen |
| IS_GET_AUTO_GAIN_MAX | Gitbt die obere Regelgrenze für AutoGain zurück |
| IS_SET_AUTO_SHUTTER_MAX | Obere Regelgrenze für AutoShutter setzen |
| IS_GET_AUTO_SHUTTER_MAX | Gibt die obere Regelgrenze für AutoShutter zurück |
| IS_SET_AUTO_SPEED | Setzt den Geschwindigkeitswert der Autofunktion. |
| IS_GET_AUTO_SPEED | Gibt den Geschwindigkeitswert der Autofunktion zurück |
| IS_SET_AUTO_WB_OFFSET | Setzt den Offset für den roten und blauen Kanal. |
| IS_GET_AUTO_WB_OFFSET | Gibt die Offsetwerte für den roten und blauen Kanal zurück. |
| IS_SET_AUTO_WB_GAIN_RANGE | Setzt die Regelgrenzen der Auto Whitebalance Funktion. |
| IS_GET_AUTO_WB_GAIN_RANGE | Gibt den Regelbereich der Auto Whitebalance Funktion zurück. |
| IS_SET_AUTO_WB_SPEED | Setzt die Geschwindigkeit der Auto Whitebalance Funktion. |
| IS_GET_AUTO_WB_SPEED | Gibt den Geschwindigkeitsbereich der Auto Whitebalance Funktion zurück. |
| IS_SET_AUTO_WB_ONCE | Setzt das automatische Abschalten der Auto Whitebalance Funktion |
| IS_GET_AUTO_WB_ONCE | Gibt den Zustand der automatischen Abschaltung zurück. |
| **pval1** | Übergabeparameter (Wert) |
| **pval2** | Übergabeparameter (Wert) |

Die Parameter *pval1* und *pval2* können, je nach verwendetem Typ des Parameters *param* unterschiedliche Werte annehmen.

**Auto Brightness Funktion**

Für die Enable Parameter nimmt die Variable (pval1) Werte von 0 (inaktiv) und 1(aktiv) an. Im Falle des Sollwerts für AutoGain/AutoShutter beschreibt der Wert einen Grauwert von 0-255. Bei den Regelgrenzen werden gültige Werte für GAIN (0-100) und SHUTTER (Belichtungszeit) eingestellt.

Bei Übergabe des Wertes 0 für den Parameter *IS_SET_AUTO_SHUTTER_MAX* wird die maximale Shutterzeit gesetzt. Falls sich diese durch Justierung des Kameratimings ändert, wird der aktualisierte Wert nachgeführt. Wird hingegen ein Wert des aktuell gültigen Schutterbereichs übergeben, bleibt dieser so lange gesetzt, bis ein neuer Wert übergeben wird. Beim Auslesen der Shutter-Regelgrenze wird der Benutzerwert oder, falls 0 übergeben wurde, die maximale Schutterzeit zurückgegeben.

**Auto Whitebalance Funktion**
Bei den Einstellungen für Offset und GainRange der Auto Whitebalance Funktion werden beide Variablen verwendet. Um den Offset für den roten und den blauen Kanal einzustellen, wird der Offset für rot als Variable *pval1* und der Offset für blau als *pval2* übergeben. Der Offset kann in einem Bereich von -50 bis +50 eingestellt werden. Bei den Regelgrenzen steht *pval1* für den minimalen und *pval2* für den maximalen Gainwert. Die Regelgrenzen lassen sich in einem Bereich von 0 bis 100 einstellen. Wenn der maximale Gainwert kleiner als das Minimum sein sollte, wird das Maximum auf den minimalen Gainwert gesetzt und umgekehrt.
Falls nur einer der Parameter *pval1/pval2* übergeben werden soll, dann ist für den jeweils anderen Parameter ein NULL-Zeiger zu übergeben.
Wenn die Einstellungen zurückgegeben werden sollen, dann werden die Werte an die mit *pval1* und *pval2* angegeben Adresse geschrieben.

**Vordefinierte Werte für Auto Features (Werte aus uEye.h ersichtlich):**

| | |
|---|---|
| IS_DEFAULT_AUTO_BRIGHT_REFERENCE | Default Sollwert für AutoGain und AutoShutter. |
| IS_MIN_AUTO_BRIGHT_REFERENCE | Minmaler Sollwert für AutoGain und AutoShutter |
| IS_MAX_AUTO_BRIGHT_REFERENCE | Maximaler Sollwert für AutoGain und AutoShutter |
| IS_DEFAULT_AUTO_SPEED | Defaultwert für Auto Geschwindigkeit. |
| IS_MIN_AUTO_SPEED | Minimaler Wert für Auto Geschwindigkeit |
| IS_MAX_AUTO_SPEED | Maximaler Wert für Auto Geschwindigkeit] |
| IS_DEFAULT_WB_OFFSET | Defaultwert für Auto Whitebalance Offset |
| IS_MIN_WB_OFFSET | Minimaler Wert für Auto Whitebalance Offset |
| IS_MAX_WB_OFFSET | Maximaler Wert für Auto Whitebalance Offset |
| IS_DEFAULT_AUTO_WB_SPEED | Defaultwert für Auto Whitebalance Geschwindigkeit. |
| IS_MIN_AUTO_WB_SPEED | Minimaler Wert für Auto Whitebalance Geschwindigkeit |
| IS_MAX_AUTO_WB_SPEED | Maximaler Wert für Auto Whitebalance Geschwindigkeit |

**Weitere Möglichkeiten, die AutoBrightness-Funktionen zu aktivieren/deaktivieren:**

Die AutoGain Funtktionalität kann mit der Funktion *is_SetHardwareGain()* aktiviert werden, wenn der Parameter *IS_SET_ENABLE_AUTO_GAIN* anstatt des ersten Wertes verwenden wird. Durch Setzen eines Wertes wird die Auto Funktionalität wieder deaktiviert (siehe auch *4.105 is_SetHardwareGain*).

Die AutoExposure Funtkionalität kann mit der Funktion *is_SetExposureTime()* aktiviert werden, wenn der Parameter *IS_SET_ENABLE_AUTO_SHUTTER* verwendet wird. Durch Setzen eines Wertes wird die Auto Funktionalität wieder deaktiviert (siehe auch *4.97 is_SetExposureTime*).

**Rückgabewert:**

IS_SUCCESS, IS_NO_SUCCESS

**Beispiele:**
Auto Gain aktivieren:
```
Double dEnable = 1;
int ret = is_SetAutoParameter (m_hCam ,IS_SET_ENABLE_AUTO_GAIN,
&dEnable, 0);
```

Helligkeitssollwert auf 128 stellen:
```
double soll = 128;
int ret = is_SetAutoParameter (m_hCam,IS_SET_AUTO_REFERENCE, &soll,
0);
```

Shutter Regelgrenze zurücklesen:

```
double maxShutter;
int ret = is_SetAutoParameter (m_hCam, IS_GET_AUTO_SHUTTER_MAX, & max-
Shutter, 0);
```

## 4.81. is_SetBadPixelCorrection

**Syntax:**

INT is_SetBadPixelCorrection (HIDS hf, INT nEnable, INT threshold)

**Beschreibung:**

*is_SetBadPixelCorrection()* schaltet die Hotpixel Korrektur ein oder aus. Dabei kann zwischen drei unterschiedlichen Varianten gewählt werden.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **nEnable** | |
| IS_BPC_DISABLE | Schaltet die Korrektur aus |
| IS_BPC_ENABLE_HARDWARE | Aktiviert die Hardware Korrektur (nur UI_141x Sensoren), Parameter *threshold* wird verwendet. |
| IS_BPC_ENABLE_SOFTWARE | Aktiviert die Software Korrektur auf Basis der im EEPROM hinterlegten Hotpixelliste. |
| IS_BPC_ENABLE_USER | Aktiviert die Software Korrektur mit benutzerdefinierten Werten. Zuvor muss die Funktion *SetBadPixelCorrectionTable()* aufgerufen werden. |
| IS_GET_BPC_MODE | Gibt den aktuellen Modus zurück. |
| IS_GET_BPC_THRESHOLD | Gibt den aktuellen Schwellwert zurück. |
| **threshold** | Wird nur bei UI_141x Sensoren in Verbindung mit dem Parameter *IS_BPC_ENABLE_HARDWARE* verwendet.<br>Beeinflusst den Grad der Korrektur. |

Es besteht die Möglichkeit die Hardwarekorrektur mit einem Softwaremodus zu verknüpfen, beide Softwarekorrekturen können aber nicht gemeinsam verwendet werden, mögliche Kombinationen sind:

1. IS_BPC_DISABLE
2. IS_BPC_ENABLE_HARDWARE
3. IS_BPC_ENABLE_SOFTWARE
4. IS_BPC_ENABLE_USER
5. IS_BPC_ENABLE_HARDWARE **|** IS_BPC_ENABLE_SOFTWARE
6. IS_BPC_ENABLE_HARDWARE **|** IS_BPC_ENABLE_USER

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*, den aktuellen Modus in Verbindung mit *IS_GET_BPC_MODE* oder den aktuelle Schwellwert in Verbindung mit *IS_GET_BPC_THRESHOLD*

## 4.82. is_SetBadPixelCorrectionTable

**Syntax:**

INT is_SetBadPixelCorrectionTable (HIDS hf, INT nMode, WORD *pList)

**Beschreibung:**

*is_SetBadPixelCorrectionTable()* setzt die Tabelle mit defekten Pixel, die bei der benutzerdefinierten Hotpixelkorrektur verwendet wird. Die Hotpixelkorrektur wird eingeschaltet. Jeder Wert in der Tabelle besteht aus einem 2-Byte WORD Datentyp. Der erste Wert gibt die Anzahl Pixel Koordinaten innerhalb der Tabelle an, als nächstes folgen die Koordinaten (zuerst X, dann Y). Eine Tabelle mit 3 defekten Pixeln muss wie folgt aufgebaut sein:

| 3 | X1 | Y1 | X2 | Y2 | X3 | Y3 |
|---|---|---|---|---|---|---|

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **nMode** | |
| IS_SET_BADPIXEL_LIST | Setzt eine neue benutzerdefinierte Liste.<br>Der Parameter pList Zeigt auf eine Liste in dem zuvor beschriebenen Format. |
| IS_GET_LIST_SIZE | Gibt die Anzahl an Pixel Koordinaten, die in der benutzerdefinierten Liste vorhanden sind zurück. |
| IS_GET_BADPIXEL_LIST | Kopiert die benutzerdefinierte Tabelle in den Parameter *pList*, der Speicher muss zuvor reserviert worden sein. |

**Beispiel:**

Auslesen der HotPixelCorrection Tabelle:

```
WORD *pList = NULL;
// Anzahl an Koordinaten in der Liste
DWORD nCount = is_SetBadPixelCorrectionTable(hf, IS_GET_LIST_SIZE,
NULL);
// Speicher für komplette Liste reservieren
pList = new WORD[ 1 + 2*nCount ];
is_SetBadPixelCorrectionTable(hf, IS_GET_BADPIXEL_LIST, pList);

// Liste wieder freigeben
delete [] pList;
```

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*, oder Anzahl an Koordinaten in der Liste, bei *IS_GET_LIST_SIZE*

## 4.83. is_SetBayerConversion

**Syntax:**

INT is_SetBayerConversion (HIDS hf, INT nMode)

**Beschreibung:**

*is_SetBayerConversion()* ermöglicht es zwischen drei verschiedenen Farbkonvertierungs-Algorithmen zu wählen, die sich in der Qualität und der benötigten Rechnerauslastung unterscheiden.

**Randbedingungen**

Funktion nur gültig für 24-Bit, 32-Bit und Y8 Farbformat.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **nMode** | |
| `IS_SET_BAYER_CV_NORMAL` | Die Standard Konvertierung wird nicht mehr unterstützt. Bei Auwahl dieses Modes wird der *BETTER*-Mode verwendet |
| `IS_SET_BAYER_CV_BETTER` | Bessere Qualität leicht erhöhte Rechner Auslastung. |
| `IS_SET_BAYER_CV_BEST` | Beste Qualität stärkste Rechner Auslastung. |
| `IS_GET_BAYER_CV_MODE` | Aktuelle Einstellung wird zurückgegeben. |

**Rückgabewert:**

In Verbindung mit *IS_GET_BAYER_CV_MODE* wird die aktuelle Einstellung gelesen, sonst *IS_SUCCESS* oder *IS_NO_SUCCESS*.

## 4.84. is_SetBinning

**Syntax:**

INT is_SetBinning (HIDS hf, INT mode)

**Beschreibung:**

Mit *is_SetBinning()* kann der Binningmodus sowohl in horizontaler als auch in vertikaler Richtung aktiviert werden. Dadurch kann die Bildgröße je Binnigrichtung halbiert oder geviertelt werden. Je nach Sensor kann bei aktiviertem Binning die Empfindlichkeit zunehmen oder die Framerate erhöht werden.
Zur gleichzeitigen Aktivierung von horizontalem und vertikalem Binning können die horizontalen und vertikalen Binningparameter durch ein logisches ODER verknüpft werden.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **mode** | |
| IS_BINNING_DISABLE | Deaktiviert das Binning. |
| IS_BINNING_2X_VERTICAL | Aktiviert zweifaches Binning in vertikaler Richtung. |
| IS_BINNING_4X_VERTICAL | Aktiviert vierfaches Binning in vertikaler Richtung. |
| IS_BINNING_2X_HORIZONTAL | Aktiviert zweifaches Binning in horizontaler Richtung. |
| IS_BINNING_4X_HORIZONTAL | Aktiviert vierfaches Binning in horizontaler Richtung. |
| IS_GET_BINNING | Gibt die aktuelle Einstellung zurück. |
| IS_GET_SUPPORTED_BINNING | Gibt die unterstützten Binning Modi zurück. |
| IS_GET_BINNING_TYPE | Der Rückgabewert gibt an, ob die Kamera farberhaltendes Binning verwendet (*IS_BINNING_COLOR*) oder nicht (*IS_BINNING_MONO*) |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS* oder die aktuellen Einstellungen bei *IS_GET_BINNING*

# 4.85. is_SetBlCompensation

**Syntax:**

INT is_SetBlCompensation (HIDS hf, INT nEnable, INT offset, INT reserved)

**Beschreibung:**

*is_SetBlCompensation()* aktiviert die Blacklevel-Kompensation, mit der die Bildqualität unter Umständen verbessert werden kann. Je nach Zeitpunkt der Änderung der Kompensation wirkt sich diese erst bei der Aufnahme des nächsten Bildes aus.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **nEnable** | |
| IS_BL_COMPENSATION_DISABLE | Schaltet die Kompensation aus |
| IS_BL_COMPENSATION_ENABLE | Aktiviert die Blacklevel-Kompensation, unter Verwendung des eingestellten Offset-Werts. |
| IS_GET_BL_COMPENSATION | Gibt den aktuellen Modus zurück. |
| IS_GET_BL_OFFSET | Gibt den aktuellen Offset zurück. |
| IS_GET_BL_DEFAULT_MODE | Gibt den Standard-Modus zurück |
| IS_GET_BL_DEFAULT_OFFSETE | Gibt den Standard-Offset zurück |
| IS_GET_BL_SUPPORTED_MODE | Gibt die unterstützten Modi zurück<br>Mögliche Werte:<br>IS_BL_COMPENSATION_ENABLE<br>Der verwendete Sensor unterstützt Blacklevel-Kompensation<br>IS_BL_COMPENSATION_OFFSET<br>Beim verwendeten Sensor kann der Offset der Blacklevel-Kompensation eingestellt werden |
| IS_IGNORE_PARAMETER | Der Parameter *nEnable* wird ignoriert. |
| **offset** | Beinhaltet den offset, der für die Kompensation verwendet wird. Gültige Werte liegen zwischen 0 und 255. |
| IS_IGNORE_PARAMETER | Der Parameter *offset* wird ignoriert. |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*, aktuelle Einstellung bei *IS_GET_BL_COMPENSATION* oder eingestellten Offset bei *IS_GET_BL_OFFSET*

## 4.86. is_SetBrightness

**Syntax:**

INT is_SetBrightness (HIDS hf, INT Bright)

**Beschreibung:**

*is_SetBrightness()* verstellt die Helligkeit des Bildes digital. Der Übergabeparameter *Bright* darf die Werte 0..255 annehmen. Die Veränderung der Helligkeit wird durch die Veränderung des Luminanzwertes durch einen, mit *Bright* einstellbaren Offset durchgeführt. *Bright* steht standardmäßig auf 100 (*IS_DEFAULT_BRIGHTNESS*).

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **Bright** | |
| 0...255 | zu setzender Helligkeitswert |
| IS_GET_BRIGHTNESS | Rücklesen der aktuellen Einstellung |

**Rückgabewert:**

Aktuelle Einstellung in Verbindung mit *IS_GET_BRIGHTNESS*, sonst *IS_SUCCESS*, *IS_NO_SUCCESS*

## 4.87. is_SetCameraID

**Syntax:**

INT is_SetCameraID (HIDS hf, INT nID)

**Beschreibung:**

*is_SetCameraID()* ermöglicht es die ID nummer zu vergeben, die zum Öffnen der Camera verwendet werden kann (siehe dazu auch is_InitCamera).

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **nID** | |
| 1..254 | Neue Kamera ID |
| IS_GET_CAMERA_ID | Gibt die aktuelle ID zurück. |

**Rückgabewert:**

In Verbindung mit *IS_GET_CAMERA_ID* wird die aktuelle ID zurückgegeben, sonst *IS_SUCCESS* oder *IS_NO_SUCCESS*.

## 4.88. is_SetColorCorrection

**Syntax:**

INT is_SetColorCorrection (HIDS hf, INT nEnable, double *factors)

**Beschreibung:**

Um eine bessere Farbwiedergabe mit Farbsensoren zu erhalten, kann über das SDK eine Farbkorrektur aktiviert werden. Hierfür steht die Funktion *is_SetColorCorrection()* zur Verfügung.

Nach Änderung des Parameters muss zwingend mit *is_SetWhiteBalance()* ein Weißabgleich durchgeführt werden (siehe *4.80 is_SetAutoParameter*).

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **nEnable** | |
| IS_CCOR_DISABLE | Deaktiviert die Farbkorrektur |
| IS_CCOR_ENABLE | Aktiviert die Farbkorrektur |
| IS_GET_CCOR_MODE | Gibt aktuelle Einstellung zurück |
| **factors** | reserviert |

**Rückgabewert:**

Aktuelle Einstellung in Verbindung mit *IS_GET_CCOR_MODE()*, sonst *IS_SUCCESS, IS_NO_SUCCESS*

## 4.89. is_SetColorMode

**Syntax:**

INT is_SetColorMode (HIDS hf, INT Mode)

**Beschreibung:**

*is_SetColorMode()* stellt den gewünschten Farbmodus ein mit dem die Bilddaten gespeichert bzw. in der VGA-Karte dargestellt werden. Im ersten Fall ist es wichtig, dass je nach verwendetem Farbmodus der allokierte Bildspeicher ausreichend groß ist. Ein 24-Bit Farbbild benötigt die dreifache Speichergröße als ein 8-Bit Monochrombild. Beim Zugriff auf die Bilddaten ist Kenntnis über die Speicherorganisation in den jeweiligen Farbmodi wichtig (siehe auch *2.4 Farb- und Speicherformate*). Eine falsche Interpretation des Speicherinhalts führt zu falschen Ergebnissen. Bei der direkten Übertragung in den Bildspeicher der VGA-Karte muss sichergestellt sein, dass die Display-Einstellungen mit den Einstellungen des Farbmodus übereinstimmen. Unter Umständen kommen die Bilder sonst in verfälschten Farben oder unkenntlich zur Darstellung.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **Mode** | |
| IS_SET_CM_RGB32 | 32-Bit Echtfarbmodus; R-G-B-Dummy |
| IS_SET_CM_RGB24 | 24-Bit Echtfarbmodus; R-G-B |
| IS_SET_CM_RGB16 | Hi-Color-Modus; 5 R - 6 G - 5 B |
| IS_SET_CM_RGB15 | Hi-Color-Modus; 5 R - 5 G - 5 B |
| IS_SET_CM_Y8 | 8-Bit Monochrombilder |
| IS_SET_CM_BAYER | Roh-Bayerdaten bei Farb-Sensoren |
| IS_SET_CM_UYVY | 16 Bit UYVY Format |
| IS_GET_COLOR_MODE | Rücklesen der aktuellen Einstellung |

**Rückgabewert:**

Aktuelle Einstellung in Verbindung mit *IS_GET_COLOR_MODE*, sonst *IS_SUCCESS*, *IS_NO_SUCCESS*

## 4.90. is_SetContrast

**Syntax:**

INT is_SetContrast (HIDS hf, INT Cont)

**Beschreibung:**

*is_SetContrast()* verändert digital den Kontrast des Bildes (Verstärkung der Luminanz) im Bereich zwischen 0% und 200%.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **Cont** | |
| 0...511 | zu setzender Kontrastwert |
| IS_GET_CONTRAST | Rücklesen der aktuellen Einstellung |

**Rückgabewert:**

Aktuelle Einstellung in Verbindung mit *IS_GET_CONTRAST*, sonst *IS_SUCCESS, IS_NO_SUCCESS*

## 4.91. is_SetConvertParam

**Syntax:**

INT is_SetConvertParam(HIDS hf, BOOL ColorCorrection, INT BayerConversionMode, INT ColorMode, INT Gamma, double *WhiteBalanceMultipliers)

**Beschreibung:**

*is_SetConvertParam()* stellt die Konvertierungsparameter für das RAW Bayer Bild ein, welches mit der Funktion *is_ConvertImage()* in ein anderes Format umgewandelt wird. Diese Funktion setzt die Werte für:

- Farbkorrektur
- Bayer Konvertierung
- Farbmodus
- Gamma
- White Balance Multipliers

**Übergabeparameter:**

| | |
|---|---|
| **Hf** | Handle auf Kamera |
| **ColorCorrection** | Aktiviert/deaktiviert die Farbkorrektur |
| **BayerConversionMode** | Setzt den Bayer Konvertierungs Modus. |
| IS_SET_BAYER_CV_BETTER | Bessere Qualität |
| IS_SET_BAYER_CV_BEST | Beste Qualität – höhere CPU Belastung |
| **ColorMode** | sets the color mode of the output image |
| IS_SET_CM_RGB32 | 32 Bit True Colour Mode; R-G-B-Dummy |
| IS_SET_CM_RGB24 | 24 Bit True Colour Mode; R-G-B |
| IS_SET_CM_RGB16 | Hi Colour Mode; 5 R - 6 G - 5 B |
| IS_SET_CM_RGB15 | Hi Colour Mode; 5 R - 5 G - 5 B |
| IS_SET_CM_Y8 | 8 Bit Monochrome Bild |
| IS_SET_CM_UYVY | 16 Bit UYVY Format |
| **Gamma** | Gammawert multipliziert mit 100. - Bereich: [1…1000] |
| **WhiteBalanceMultipliers** | Zeiger auf ein Array mit der Rot-, Grün- und Blauverstärkung |

**Rückgabewert:**

IS_SUCCESS, IS_NO_SUCCESS, IS_INVALID_COLOR_FORMAT oder IS_INVALID_PARAMETER

**Beispiel:**

Siehe *4.6 is_ConvertImage*

## 4.92. is_SetDDUpdateTime

**Syntax:**

INT is_SetDDUpdateTime (HIDS hf, INT ms)

**Beschreibung:**

*is_SetDDUpdateTime()* setzt das Timer-Intervall für den Update-Zyklus des Videobildes im DirectDraw BackBuffer-Modus. Gültige Werte sind 20ms bis 2000ms.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **ms** | Zeit in Millisekunden |

**Rückgabewert:**

IS_SUCCESS, IS_NO_SUCCESS

 

## 4.93. is_SetDisplayMode

**Syntax:**

INT is_SetDisplayMode (HIDS hf, INT Mode)

**Beschreibung:**

*is_SetDisplayMode()* bestimmt auf welche Art die Bilder auf dem Bildschirm dargestellt werden. Für echtes Live-Video plus Overlay wurde der DirectDraw Overlay-Surface-Modus eingeführt. Die Verfügbarkeit dieses Modus ist abhängig von der verwendeten VGA-Karte. Nur bestimmte VGA Controller unterstützen diesen Modus.
Der Speicherausbau der VGA-Karte sollte genügend groß sein, da der Overlay-Modus einen Speicherplatz bis zur Größe der aktuellen Bildschirm-Auflösung für sich benötigt.

Im Allgemeinen ist das uEye SDK *thread safe*. Die API-Funktionsaufrufe erfolgen alle in *critical sections*. Auf Grund interner *DirectDraw* Strukturen empfehlen wird dringend, die folgenden Funktionen nur aus einem *Thread* heraus aufzurufen, um ein unvorhersehbares Verhalten Ihrer Applikation zu vermeiden.

- *is_InitCamera()*
- *is_SetDisplayMode()*
- *is_ExitCamera()*

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **Mode** | |
| IS_SET_DM_DIB | Bild in Systemspeicher (RAM) erfassen (keine automatische Darstellung – Darstellung mit *is_RenderBitmap()* möglich!) |
| IS_SET_DM_DIRECTDRAW \| IS_SET_DM_BACKBUFFER | DirectDraw BackBuffer Modus |
| IS_SET_DM_DIRECTDRAW \| IS_SET_DM_ALLOW_OVERLAY | DirectDraw Overlay Surface Modus |
| **DirectDraw Overlay Surface Erweiterung** | |
| IS_SET_DM_ALLOW_SCALING | Echtzeit-Skalierung im Overlay Surface Modus |
| **Rücklesemodus** | |
| IS_GET_DISPLAY_MODE | Rücklesen der aktuellen Einstellung |

**Rückgabewert:**

Aktuelle Einstellung in Verbindung mit *IS_GET_DISPLAY_MODE*, sonst *IS_SUCCESS, IS_NO_SUCCESS*

**Beispiel**
Auflösung mit 1024x768x16 = 1,5 MB -> OverlayBuffer bis zu 1,5 MB

Beispiel: is_SetDisplayMode (hf, Mode);

Bitmap-Modus (in Systemspeicher digitalisieren):
```
Mode = IS_SET_DM_DIB
```

DirectDraw BackBuffer Modus
```
Mode = IS_SET_DM_DIRECTDRAW
```

DirectDraw Overlay-SurfaceModus (bestes Live-Overlay):
```
Mode =  IS_SET_DM_DIRECTDRAW  |
  IS_SET_DM_ALLOW_OVERLAY
```
bzw. um ein automatisches Skalieren auf die Fenstergröße zu erlauben:
```
Mode =  IS_SET_DM_DIRECTDRAW  |
  IS_SET_DM_ALLOW_OVERLAY  |
  IS_SET_DM_ALLOW_SCALING
```

## 4.94. is_SetDisplayPos

**Syntax:**

INT is_SetDisplayPos (HIDS hf, INT x, INT y)

**Beschreibung:**

Die Funktion *is_SetDisplayPos()* ermöglicht die Verschiebung der Bildausgabe, die mit is_RenderBitmap erzeugt wird. Die Verschiebung erfolgt über die Parameter *x* und *y*.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **x** | Offset in x-Richtung |
| **y** | Offset in y-Richtung |

**Rückgabewert:**

IS_SUCCESS, IS_NO_SUCCESS

## 4.95. is_SetEdgeEnhancement

**Syntax:**

INT is_SetEdgeEnhancement (HIDS hf, INT nEnable)

**Beschreibung:**

Bedingt durch die Farbkonvertierung des Bayerformats können die ursprünglichen Kanten innerhalb eines Farbbildes leicht unscharf werden. Durch aktivieren des digitalen Kantenfilters, kann diesem Effekt entgegen gewirkt werden. Hierfür stehen zwei unterschiedlich starke Einstellungen (*IS_EDGE_EN_STRONG, IS_EDGE_EN_WEAK*) zur Verfügung. Bei Verwendung dieser Funktion erhöht sich die CPU-Belastung des Systems.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **nEnable** | |
| IS_EDGE_EN_DISABLE | Deaktiviert den Kantenfilter |
| IS_EDGE_EN_STRONG | Aktiviert die starke Kantenbetonung |
| IS_EDGE_EN_WEAK | Aktiviert die schwächere Kantenbetonung |
| IS_GET_EDGE_ENHANCEMENT | Gibt die aktuelle Einstellung zurück |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.96. is_SetErrorReport

**Syntax:**

INT is_SetErrorReport (HIDS hf, INT Mode)

**Beschreibung:**

Schaltet die Protokollierung der Fehler ein und aus. Bei eingeschaltetem Fehlerreport werden auftretende Fehler automatisch über eine Dialogbox angezeigt. Quittiert man die Dialogbox mit *Abbrechen*, wird gleichzeitig der Fehlerreport deaktiviert. Ist der Fehlerreport nicht aktiv, können Fehler in Verbindung mit der Funktion *is_GetError()* abgefragt werden. Das Kamera-Handle wird nicht ausgewertet, *is_SetErrorReport()* arbeitet global, nicht devicebezogen. *is_SetErrorReport()* kann vor *is_InitCamera()* aufgerufen werden.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera oder NULL |
| **Mode** | |
| IS_DISABLE_ERR_REP | Ausschalten der Fehleranzeige |
| IS_ENABLE_ERR_REP | Einschalten der Fehleranzeige |

**Rückgabewert:**

Aktuelle Einstellung in Verbindung mit *IS_GET_ERR_REP_MODE*, sonst *IS_SUCCESS, IS_NO_SUCCESS*

## 4.97. is_SetExposureTime

**Syntax:**

INT is_SetExposureTime (HIDS hf, double EXP, double* newEXP)

**Beschreibung:**

Die Funktion *is_SetExposureTime()* setzt die mit *EXP* angegebene Belichtungsdauer in ms. Da diese aber nur in Vielfachen einer Zeilendauer einstellbar ist, kann die tatsächlich verwendete Zeit von dem gewünschten Wert abweichen. Die tatsächlich nach Aufruf dieser Funktion eingestellte Dauer wird über den Parameter newEXP zurückgegeben. Durch Ändern der Fenstergröße oder des Auslesetimings (Pixelclock) wird die zuvor gesetzte Exposure-Zeit ebenfalls verändert, deshalb muss *is_SetExposureTime()* danach neu aufgerufen werden.
Exposure-Zeit beeinflussende Funktionen:
*is_SetImageSize()*
*is_SetPixelClock()*
*is_SetFrameRate()* (Nur wenn neue Bilddauer kürzer als Exposure-Zeit wird)

Welche minimalen- und maximalen Werte möglich sind und die Abhängigkeiten der einzelnen Sensoren wird in der Beschreibung zum uEye Timing ausführlich behandelt.
Je nach Zeitpunkt der Änderung der Belichtungszeit wirkt sich diese erst bei der Aufnahme des nächsten Bildes aus.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **EXP** | Neue gewünschte Exposure-Zeit. |
| | Wenn EXP = 0.0 übergeben wird, dann wird mit der Belichtungszeit von 1/Framerate belichtet |
| IS_GET_EXPOSURE_TIME | Gibt nur die aktuelle Exposure-Zeit über den Parameter newEXP zurück. |
| IS_GET_DEFAULT_EXPOSURE | Gibt die Standard Exposure-Zeit zurück |
| **newEXP** | Gibt die tatsächlich eingestellte Exposure-Zeit zurück. |

Bei Verwendung der Konstanten *IS_SET_ENABLE_AUTO_SHUTTER* für den Parameter *EXP* wird die AutoExposure Funktionalität aktiviert. Durch Setzen eines Wertes wird diese wieder deaktiviert (siehe auch *4.80 is_SetAutoParameter*).

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.98. is_SetExternalTrigger

**Syntax:**

INT is_SetExternalTrigger (HIDS hf, INT nTriggerMode)

**Beschreibung:**

*is_SetExternalTrigger()* aktiviert den Triggereingang. Mit dem Funktionsaufruf wird die Flanke, auf die ein Trigger ausgelöst werden soll, angegeben. Wurde der Trigger aktiviert, wird bei jedem Aufruf der Funktion *is_FreezeVideo()* mit der Bildaufnahme gewartet, bis das entsprechende Triggerereignis stattgefunden hat.

- Aktivität auf Flanke High-Low (TTL)    IS_SET_TRIG_HI_LO
- Aktivität auf Flanke Low-High (TTL)    IS_SET_TRIG_LO_HI
- Deaktivieren    IS_SET_TRIG_OFF

Wenn auf die Triggerfunktionalität verzichtet wird (IS_SET_TRIG_OFF), kann der Pegel am Triggereingang statisch abgefragt werden. Damit wird der Triggereingang als digitaler Eingang verwendet.

> Beim Kameramodell UI-144x-xx muss - bedingt durch das Timingverhalten dieses Modells - im Triggermodus die Belichtungszeit auf den Wert 1/Framerate eingestellt werden.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **nTriggerMode** | |
| IS_SET_TRIG_OFF | Ausschalten der Triggerverarbeitung |
| IS_SET_TRIG_HI_LO | Aktive Triggerflanke auf negative Signalflanke setzen |
| IS_SET_TRIG_LO_HI | Aktive Triggerflanke auf positive Signalflanke setzen |
| IS_SET_TRIG_SOFTWARE | Aktiviert Softwaretriggermodus; mit Aufruf der Funktion *is_FreezeVideo()* wird die Kamera getriggert und liefert ein Bild. |
| IS_GET_EXTERNALTRIGGER | Triggermodus-Einstellung zurück lesen |
| IS_GET_TRIGGER_STATUS | Gibt den aktuellen Pegel am Triggereingang zurück. |
| IS_GET_SUPPORTED_TRIGGER_MODE | Gibt die unterstützten Triggermodi zurück. |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS* oder aktuelle Einstellung in Verbindung mit *IS_GET_EXTERNALTRIGGER*
*IS_SET_TRIG_SOFTWARE* | *IS_SET_TRIG_HI_LO* | *IS_SET_TRIG_LO_HI* bei Vewendung von *IS_GET_SUPPORTED_TRIGGER_MODE*

**Beispiel:**

Triggermodus einschalten und *High-Active* Blitzmodus setzen.

```
is_SetExternalTrigger (hf, IS_SET_TRIG_SOFTWARE);
is_SetFlashStrobe (hf, IS_SET_FLASH_HI_ACTIVE);
is_FreezeVideo (hf, IS_WAIT);
```

## 4.99. is_SetFlashDelay

**Syntax:**

INT is_SetFlashDelay (HIDS hf, ULONG ulDelay, ULONG ulDuration)

**Beschreibung:**

*is_SetSetFlashDelay()* erlaubt es eine Verzögerung einzustellen um die der Blitz später eingeschaltet wird. Zusätzlich kann die Dauer des Blitzes angegeben werden. So kann eine globale Blitzfunktion realisiert werden, in der alle Zeilen eines Rolling-Shutter Sensors belichtet werden. (siehe dazu auch *4.35 is_GetGlobalFlashDelays*).
Wird für *ulDelay* der Wert 0 angegeben, wird die normale Blitzfunktion verwendet.
Soll nur verspätet eingeschaltet werden, und erst bei Bildende ausgeschaltet werden, kann für die Blitzdauer *ulDuration* der Wert 0 übergeben werden.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **ulDelay** | Zeit um die der Blitz verzögert wird (in µs) |
| IS_GET_FLASH_DELAY | Rückgabe der momentan eingestellten Verzögerung. |
| IS_GET_FLASH_DURATION | Rückgabe der momentan eingestellten Blitzdauer. |
| IS_GET_MIN_FLASH_DELAY | Gibt den minimal einstellbaren Wert für die Verzögerung zurück. |
| IS_GET_MIN_FLASH_DURATION | Gibt den minimal einstellbaren Wert für die Blitzdauer zurück. |
| IS_GET_MAX_FLASH_DELAY | Gibt den maximal einstellbaren Wert für die Verzögerung zurück. |
| IS_GET_MAX_FLASH_DURATION | Gibt den maximal einstellbaren Wert für die Blitzdauer zurück. |
| IS_GET_FLASH_DELAY_GRANULARITY | Gibt die Auflösung der einstellbaren Verzögerungszeit zurück. |
| IS_GET_FLASH_DURATION_GRANULARITY | Gibt die Auflösung der einstellbaren Blitzdauer zurück. |
| **ulDuration** | Zeit in der der Blitz eingeschaltet wird (in µs) |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*, aktuelle Einstellungen in Verbindung mit *IS_GET_FLASH_DELAY* oder *IS_GET_FLASH_DURATION*.

## 4.100. is_SetFlashStrobe

**Syntax:**

INT is_SetFlashStrobe (HIDS hf, INT nMode, INT nLine)

**Beschreibung:**

*is_SetFlashStrobe()* schaltet den Flash-Strobe ein. Dabei kann über den Parameter *nMode die Ansteuerung aktiviert oder deaktiviert werden. Außerdem lässt sich der aktive Level (high oder low) setzen.* Per Default ist der Strobe high-aktiv.
Mit den Konstanten *IS_SET_FLASH_HIGH* und *IS_SET_FLASH_LOW* kann der Strobe-Ausgang als digitaler Ausgang genutzt werden.
Die Dauer des Blitzes und die Auslöseverzögerung kann über die Funktion *is_SetFlashDelay()* (siehe *4.99 is_SetFlashDelay*) eingestellt werden. Für Kameras mit Rolling Shutter Sensoren wird empfohlen, die Werte aus der Funktion *is_GetGlobalFlashDelays()* (siehe *4.35 is_GetGlobalFlashDelays*) als Auslöseverzögerung und Blitzdauer zu verwenden, da es sonst zu unerwarteten Ergebnissen kommen kann. Beim Blitzen im Capturemodus ist hierauf besonders zu achten.
Für die Modi *high-aktiv* und *low-aktiv* muss der jeweilige Parameter passend zum Kameramodus gewählt werden.
*IS_SET_FLASH_LO_ACTIVE* und *IS_SET_FLASH_HI_ACTIVE* werden für den Triggermodus (siehe *4.98 is_SetExternalTrigger*) benötigt.
*IS_SET_FLASH_LO_ACTIVE_FREERUN* und *IS_SET_FLASH_HI_ACTIVE_FREERUN* funktioniert nur im Capturemodus (siehe *4.4 is_CaptureVideo*).

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **nMode** | |
| IS_SET_FLASH_OFF | Schaltet den Strobe-Ausgang aus. |
| IS_SET_FLASH_LO_ACTIVE | Schaltet den Strobe-Ausgang auf Low-Active. |
| IS_SET_FLASH_HI_ACTIVE | Schaltet den Strobe-Ausgang auf High-Active. |
| IS_SET_FLASH_LO_ACTIVE_FREERUN | Schaltet den Strobe-Ausgang auf Low-Active im Freerun. |
| IS_SET_FLASH_HI_ACTIVE_FREERUN | Schaltet den Strobe-Ausgang auf High-Active im Freerun. |
| IS_SET_FLASH_HIGH | Setzt Strobe-Ausgang auf HIGH. |
| IS_SET_FLASH_LOW | Setzt Strobe-Ausgang auf LOW. |
| IS_GET_FLASHSTROBE_MODE | Gibt den aktuellen Modus zurück. |
| IS_SET_FLASH_IO_1 | Blitzen auf dem I/O-Port 1 (nur uEyeLE) |
| IS_SET_FLASH_IO_2 | Blitzen auf dem I/O-Port 2 (nur uEyeLE) |
| IS_GET_SUPPORTED_FLASH_IO | Gibt die unterstützten, blitzfähigen I/O-Ports zurück (nur uEyeLE) |
| **nLine** | Wird aktuell nicht verwendet. |

⚠ Die Parameter *IS_SET_FLASH_LO_ACTIVE_FREERUN* und *IS_SET_FLASH_HI_ACTIVE_FREERUN* werden von den Kameras UI-1440 und UI-1210 nicht unterstützt.

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*, den aktuellen Modus in Verbindung mit *IS_GET_FLASHSTROBE_MODE*

**Beispiel:**

Triggermodus einschalten und *High-Active* Blitzmodus setzen.

```
is_SetExternalTrigger (hf, IS_SET_TRIG_SOFTWARE);
is_SetFlashStrobe (hf, IS_SET_FLASH_HI_ACTIVE, 0);
is_FreezeVideo (hf, IS_WAIT);
```

## 4.101. is_SetFrameRate

**Syntax:**

INT is_SetFrameRate (HIDS hf, double FPS, double* newFPS)

**Beschreibung:**

Durch Aufruf der Funktion *is*_SetFrameRate*()* kann die gewünschte Anzahl an Bildern pro Sekunde, mit der der Sensor arbeitet gesetzt werden. Wird die Framerate zu hoch eingestellt, kann nicht mehr jedes Bild eingelesen werden, wodurch die tatsächliche Framerate absinkt. Da hier ebenfalls wie bei der Exposure-Zeit nicht jeder beliebige Wert eingestellt werden kann, wird nach Aufruf dieser Funktion die neue Framerate über den Parameter newFPS zurückgegeben. Genauere Details hierzu befinden sich in der Beschreibung zum uEye Timing.
Ähnlich wie bei der Exposure-Zeit wirken sich Änderungen an der Fenstergröße oder an dem Pixelclock auf die Framerate aus.
Die Framerate beeinflussende Funktionen:

- *is_SetImageSize()*
- *is_SetPixelClock()*

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **FPS** | Gewünschte Anzahl an Bildern pro Sekunde. |
| IS_GET_FRAMERATE | Gibt nur die aktuelle Framerate über den Parameter newFPS zurück. |
| IS_GET_DEFAULT_FRAMERATE | Gibt den Standard-Framerate zurück |
| **newFPS** | Gibt die tatsächlich eingestellte Framerate zurück. |

**Rückgabewert:**

*IS_SUCCESS oder IS_NO_SUCCESS*

## 4.102. is_SetGainBoost

**Syntax:**

INT is_SetGainBoost (HIDS hf, INT mode)

**Beschreibung:**

*is_* SetGainBoost*()* aktiviert oder deaktiviert die zusätzliche Hardwareverstärkung der Kamera. Diese Funktion wird von den folgenden Kameramodellen untersützt: UI-1220-C/M, UI-1440-C/M, UI-1540-C/M, UI-1450-C, UI-1460-C, UI-1480-C und monochrome CCD-Modelle.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **mode** | |
| IS_GET_GAINBOOST | Gibt den aktuellen Zustand der Zusatzverstärkung zurück. Falls die Kamera diese Funktion nicht unterstützt, wird *IS_NOT_SUPPORTED* zurückgegeben. |
| IS_SET_GAINBOOST_ON | Aktiviert die Zusatzverstärkung |
| IS_SET_GAINBOOST_OFF | Deaktiviert die Zusatzverstärkung |
| IS_GET_SUPPORTED_GAINBOOST | Gibt *IS_SET_GAINBOOST_ON* zurück, falls die Funktion unterstützt wird, sonst *IS_SET_GAINBOOST_OFF*. |

**Rückgabewert:**

Bei Aufruf mit I*S_GET_GAINBOOST* die aktuellen Einstellungen, sonst *IS_NOT_SUPPORTED*, *IS_SUCCESS* oder *IS_NO_SUCCESS*

## 4.103. is_SetGamma

**Syntax:**

INT is_SetGamma (HIDS hf, INT Gamma)

**Beschreibung:**

*is_SetGamma()* setzt den Wert für die digitale Gamma Korrektur. Der gültige Wertebereich liegt zwischen 0.01 und 10. Allerdings muss der Parameter *Gamma* als Integer-Wert mit einem Bereich von 1 bis 1000 (Gammawert * 100) festgelegt sein.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **Gamma** | Gammawert multipliziert mit 100. - Bereich: [1…1000] |
| IS_GET_GAMMA | Rücklesen der aktuellen Einstellung |

**Rückgabewert:**

Aktuelle Einstellung in Verbindung mit *IS_GET_BRIGHTNESS*, sonst *IS_SUCCESS* or *IS_NO_SUCCESS*

**Beispiel:**

Gammawert auf 1.42 setzen:

```
ret = SetGamma(hf, 142);
```

## 4.104. is_SetGlobalShutter

**Syntax:**

INT is_SetGlobalShutter (HIDS hf, INT mode)

**Beschreibung:**

*is*_SetGlobalShutter*()* aktiviert/deaktiviert den *Global Start Shutter*. Diese Funktionnalität wird nur von der UI-1480-C unterstützt.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **mode** | |
| IS_GET_GLOBAL_SHUTTER | Gibt den aktuellen Global Shutter Modus zurück oder *IS_NOT_SUPPORTED,* wenn die Kamera diese Funktion nicht unterstützt. |
| IS_SET_ GLOBAL_SHUTTER_ON | Aktiviert den Global Shutter |
| IS_SET_ GLOBAL_SHUTTER_OFF | Deaktiviert den Global Shutter |
| IS_GET_SUPPORTED_GLOBAL_SHUTTER | Gibt *IS_SET_*GLOBAL_SHUTTER*_ON* zurück, falls diese Funktion unterstützt wird. Sonst wird *IS_SET_*GLOBAL_SHUTTER*_OFF* zurück gegeben. |

**Rückgabewert:**

Aktuelle Einstellung beim Aufruf mit *IS_GET_GLOBAL_SHUTTER* sonst *IS_NOT_SUPPORTED*, *IS_SUCCESS* oder *IS_NO_SUCCESS*

 

## 4.105. is_SetHardwareGain

**Syntax:**

INT is_SetHardwareGain (HIDS hf, INT nMaster, INT nRed, INT nGreen, INT nBlue)

**Beschreibung:**

Die Funktion *is_SetHardwareGain()* dient zum Steuern der in der Kamera befindlichen Verstärker, die unabhängig voneinander von 0% bis 100% eingestellt werden können. Über die Funktion *is_GetSensorInfo()* kann abgefragt werden welche Verstärker vorhanden sind.

Je nach Zeitpunkt der Änderung der Verstärkung wirkt sich diese erst bei der Aufnahme des nächsten Bildes aus.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **nMaster** | Gesamt Verstärkung. |
| IS_IGNORE_PARAMETER | Master-Verstärkung nicht verändern. |
| IS_GET_MASTER_GAIN | Rückgabe der Master-Verstärkung |
| IS_GET_RED_GAIN | Rückgabe der Rot-Verstärkung |
| IS_GET_GREEN_GAIN | Rückgabe der Grün-Verstärkung |
| IS_GET_BLUE_GAIN | Rückgabe der Blau-Verstärkung |
| IS_GET_DEFAULT_MASTER | Rückgabe der Standard-Master-Verstärkung |
| IS_GET_ DEFAULT_RED | Rückgabe der Standard-Rot-Verstärkung |
| IS_GET_ DEFAULT_GREEN | Rückgabe der Standard-Grün-Verstärkung |
| IS_GET_ DEFAULT_BLUE | Rückgabe der Standard-Blau-Verstärkung |
| **nRed** | Rotkanal |
| IS_IGNORE_PARAMETER | Rot nicht verändern. |
| **nGreen** | Grünkanal |
| IS_IGNORE_PARAMETER | Grün nicht verändern. |
| **nBlue** | Blaukanal |
| IS_IGNORE_PARAMETER | Blau nicht verändern. |

> Bei Verwendung der Konstanten *IS_SET_ENABLE_AUTO_GAIN* für den Parameter EXP wird die AutoGain Funktionalität aktiviert. Durch Setzen eines Wertes wird diese wieder deaktiviert (siehe auch *4.80 is_SetAutoParameter*).

**Rückgabewert:**

Aktuelle Einstellung in Verbindung mit *IS_GET_MASTER_GAIN, IS_GET_RED_GAIN, IS_GET_ GREEN_GAIN, IS_GET_BLUE_GAIN* sonst *IS_SUCCESS* oder *IS_NO_SUCCESS*

## 4.106. is_SetHardwareGamma

**Syntax:**

INT is_SetHardwareGamma (HIDS hf, INT nMode)

**Beschreibung:**

Die Funktion *is_SetHardwareGamma()* aktiviert bzw. deaktiviert die Gammaregelung der Kamera.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **nMode** | |
| IS_GET_HW_SUPPORTED_GAMMA | IS_SET_HW_GAMMA_ON<br>Die Gammaregelung wird von der Kamera unterstützt.<br>IS_SET_HW_GAMMA_OFF<br>Die Gammaregelung wird von der Kamera nicht unterstützt. |
| IS_SET_HW_GAMMA_OFF | Aktiviert die Gammaregelung. |
| IS_SET_HW_GAMMA_ON | Deaktiviert die Gammaregelung. |
| IS_GET_HW_GAMMA | Gibt den aktuellen Zustand der Gammaregelung zurück. |

**Rückgabewert:**

*IS_SUCCESS*, *IS_NO_SUCCESS* oder *IS_NOT_SUPPORTED*

## 4.107. is_SetHWGainFactor

**Syntax:**

INT is_SetHWGainFactor (HIDS hf, INT nMode, INT nFactor)

**Beschreibung:**

Die Funktion *is_SetHWGainFactor()* dient der Steuerung der in der Kamera befindlichen Verstärker. Diese können unabhängig voneinander, von einfacher bis maximaler Verstärkung, eingestellt werden. Über die Funktion *is_GetSensorInfo()* kann abgefragt werden, welche Verstärker vorhanden sind.
Für die Umrechnung eines Verstärkungswertes aus der Funktion *is_SetHardwareGain()* kann der Parameter *nMode* auf einen der *IS_INQUIRE_x_FACTOR* Werte gesetzt werden. Der Wertebereich für *nFactor* liegt in diesem Fall von 0 bis 100.
Der Parameter *nFactor* muss für das Setzen der Verstärkung mit *IS_GET_x_GAIN_FACTOR* als Integer-Wert mit einem Bereich von 100 bis Maximum festgelegt sein. Das Maximum kann mit *IS_INQUIRE_x_FACTOR* und einem Wert von 100 für *nFactor* abgefragt werden. Ein Verstärkungsfaktor mit dem Wert 100 entspricht keiner Verstärkung, ein Wert von 200 einem Verstärkungsfaktor von zwei, usw..
Der Rückgabewert entspricht der gesetzen Verstärkung. Diese kann von der gewünschten abweichen, da nur bestimmte Werte gesetzt werden können. Es wird immer der Verstärkungswert gesetzt, der der Vorgabe am nächsten liegt.
Je nach Zeitpunkt der Änderung der Verstärkung wirkt sich diese erst bei der Aufnahme des nächsten Bildes aus.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **nMode** | |
| IS_GET_MASTER_GAIN_FACTOR | Rückgabe der Master-Verstärkung |
| IS_GET_RED_GAIN_FACTOR | Rückgabe der Rot-Verstärkung |
| IS_GET_GREEN_GAIN_FACTOR | Rückgabe der Grün-Verstärkung |
| IS_GET_BLUE_GAIN_FACTOR | Rückgabe der Blau-Verstärkung |
| IS_SET_MASTER_GAIN_FACTOR | Setzen der Master-Verstärkung |
| IS_SET_RED_GAIN_FACTOR | Setzen der Rot-Verstärkung |
| IS_SET_GREEN_GAIN_FACTOR | Setzen der Grün-Verstärkung |
| IS_SET_BLUE_GAIN_FACTOR | Setzen der Blau-Verstärkung |
| IS_GET_DEFAULT_MASTER_GAIN_FACTOR | Rückgabe der Standard-Master-Verstärkung |
| IS_GET_DEFAULT_RED_GAIN_FACTOR | Rückgabe der Standard-Rot-Verstärkung |
| IS_GET_DEFAULT_GREEN_GAIN_FACTOR | Rückgabe der Standard-Grün-Verstärkung |
| IS_GET_DEFAULT_BLUE_GAIN_FACTOR | Rückgabe der Standard-Blau-Verstärkung |
| IS_INQUIRE_MASTER_GAIN_FACTOR | Umrechnung des Index Wertes für die Master-Verstärkung |
| IS_INQUIRE_RED_GAIN_FACTOR | Umrechnung des Index Wertes für die Rot-Verstärkung |
| IS_INQUIRE_GREEN_GAIN_FACTOR | Umrechnung des Index Wertes für die Grün-Verstärkung |
| IS_INQUIRE_BLUE_GAIN_FACTOR | Umrechnung des Index Wertes für die Blau-Verstärkung |

| | |
|---|---|
| **nFactor** | Verstärkungswert |

**Rückgabewert:**

Aktuelle Einstellung in Verbindung mit *IS_GET_MASTER_GAIN_FACTOR, IS_GET_RED_GAIN_FACTOR, IS_GET_ GREEN_GAIN_FACTOR, IS_GET_BLUE_GAIN_FACTOR.*

Gesetzte Einstellung nach Verwendung von *IS_SET_MASTER_GAIN_FACTOR, IS_SET_RED_GAIN_FACTOR, IS_SET_ GREEN_GAIN_FACTOR, IS_SET_BLUE_GAIN_FACTOR.*

Standard Einstellung nach Verwendung von *IS_GET_DEFAULT_MASTER_GAIN_FACTOR, IS_GET_DEFAULT_RED_GAIN_FACTOR, IS_GET_DEFAULT_ GREEN_GAIN_FACTOR, IS_GET_DEFAULT_BLUE_GAIN_FACTOR.*

Umgerechneten Verstärkungsindex nach Verwendung von *IS_INQUIRE_ MASTER_GAIN_FACTOR, IS_INQUIRE_ RED_GAIN_FACTOR, IS_INQUIRE _ GREEN_GAIN_FACTOR, IS_INQUIRE _BLUE_GAIN_FACTOR.*

**Beispiel:**

Masterverstärkungsfaktor auf 3.57 setzen:

```
ret = is_SetHWGainFactor(hf, IS_SET_MASTER_GAIN_FACTOR, 357);
//ret hat für die UI-1460-C den Wert 363
```

Abfragen des maximalen Verstärkungsfaktors im Rotkanal:

```
ret = is_SetHWGainFactor(hf, IS_INQUIRE_RED_GAIN_FACTOR, 100);
//ret hat für die UI-1460-C den Wert 725
```

## 4.108. is_SetHwnd

**Syntax:**

INT is_SetHwnd (HIDS hf, HWND hwnd)

**Beschreibung:**

*is_SetHwnd()* setzt ein neues Fenster Handle für die Bildausgabe unter DirectDraw. Das neue Handle und die Bildausgabe wird erst beim nächsten Aufruf von *is_SetDisplayMode()* wirksam.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **hwnd** | Handle auf ein Fenster |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.109. is_SetImageAOI

**Syntax:**

INT is_SetImageAOI (HIDS hf, INT xPos, INT yPos, INT width, INT height)

*is_SetImageAOI()* wird vollständigt durch die Funktion *is_SetAOI()* ersetzt (siehe *4.79 is_SetAOI*)

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **xPos** | x-Position der oberen linken Ecke |
| **yPos** | y-Position der oberen linken Ecke |
| **width** | Bildbreite |
| **height** | Bildhöhe |

**Rückgabewert:**

IS_SUCCESS oder IS_NO_SUCCESS.

## 4.110. is_SetImageMem

**Syntax:**

INT is_SetImageMem (HIDS hf, char* pcImgMem, INT id)

**Beschreibung:**

*is_SetImageMem()* setzt den angegebenen Bildspeicher als aktiven Speicher. Nur ein aktiver Bildspeicher kann Bilddaten empfangen. Nach *is_SetImageMem()* muss ein *is_SetImageSize()* folgen, um die Bildgegebenheiten auf den neu aktivierten Speicher zu übertragen. Für *pcImg-Mem* muss ein von *is_AllocImgMem()* stammender Zeiger übergeben werden.
Alle anderen Zeiger führen zu einer Fehlermeldung! Eine wiederholte Übergabe desselben Zeigers ist erlaubt.

In den DirectDraw Modi ist das Setzen eines Bildspeichers nicht erforderlich!

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **pcImgMem** | Zeiger auf den Speicheranfang |
| **id** | ID für diesen Speicher |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.111. is_SetImagePos

**Syntax:**

INT is_SetImagePos (HIDS hf, INT x, INT y)

**Beschreibung:**

*is_SetImagePos()* bestimmt den Startpunkt des Fensters. In Verbindung mit der Funktion *is_SetImageSize()* kann aus dem vollen Videobild ein Ausschnitt herausgeschnitten werden. Dabei ist die nachfolgend dargestellte Reihenfolge der Funktionsaufrufe zwingend erforderlich:

1. *is_SetImageSize()*
2. *is_SetImagePos()*

Die Funktion *is_SetAOI()* (siehe *4.79 is_SetAOI*) vereint die beiden Funktionen. Mit *is_SetAOI()* können die Position und die Größe eines AOI mit einem Funktionsaufruf gesetzt werden.
Die Parameter x und y stellen einen Offset in Bezug auf die linke obere Ecke des Bildes dar. Das ausgeschnittene Fenster wird dabei an den Anfang des Speichers kopiert. Soll das Bild an denselben Offset innerhalb des Speichers kopiert werden, kann die neue Position mit den Parametern *IS_SET_IMAGE_POS_X_ABS* und *IS_SET_IMAGE_POS_Y_ABS* logisch Oder verknüpft werden.

**Beispiel:**

x = 20
y = 20

x = 20 | IS_SET_IMAGE_POS_X_ABS
y = 20

x = 20 | IS_SET_IMAGE_POS_X_ABS
y = 20 | IS_SET_IMAGE_POS_Y_ABS

Sensor Bildfläche
Speicher Abbild

**Randbedingungen:**

Siehe *4.79 is_SetAOI*.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **x** | |
| 0...xMax | Horizontale Position |
| IS_GET_IMAGE_POS_X | Rücklesen der aktuellen X-Position |
| IS_GET_IMAGE_POS_X_ABS | Rücklesen ob die X-Position für den aktuellen Speicher übernommen wurde |
| IS_GET_IMAGE_POS_X_MIN | Kleinster Wert für die horizontale AOI-Position |
| IS_GET_IMAGE_POS_X_MAX | Größter Wert für die horizontale AOI-Position |
| IS_GET_IMAGE_POS_X_INC | Schrittweite für die horizontale AOI-Position |
| IS_GET_IMAGE_POS_Y | Rücklesen ob die Y-Position für den aktuellen Speicher übernommen wurde |
| IS_SET_IMAGE_POS_Y_ABS | Position absolut auch für Speicher übernehmen |
| IS_GET_IMAGE_POS_Y_MIN | Kleinster Wert für die vertikale AOI-Position |
| IS_GET_IMAGE_POS_Y_MAX | Größter Wert für die vertikale AOI-Position |
| IS_GET_IMAGE_POS_Y_INC | Schrittweite für die vertikale AOI-Position |
| **y** | |
| 0...yMax | Vertikale Position |

**Rückgabewert:**

Aktuelle Einstellung in Verbindung mit *IS_GET_IMAGE_POS_X* und *IS_GET_IMAGE_POS_Y* als Übergabeparameter für x, sonst *IS_SUCCESS, IS_NO_SUCCESS*

## 4.112. is_SetImageSize

**Syntax:**

INT is_SetImageSize (HIDS hf, INT x, INT y)

**Beschreibung:**

*is_SetImageSize()* bestimmt in Verbindung mit den Einstellungen durch *is_SetImagePos()* die Bildgröße.
Die nachfolgend dargestellte Reihenfolge der Funktionsaufrufe ist zwingend erforderlich:

1. *is_SetImageSize()*
2. *is_SetImagePos()*

Die Funktion *is_SetAOI()* (siehe *4.79 is_SetAOI*) vereint die beiden Funktionen. Mit *is_SetAOI()* können die Position und die Größe eines AOI mit einem Funktionsaufruf gesetzt werden.

**Randbedingungen:**

Siehe *4.79 is_SetAOI*.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **x** | |
| 1...xMax | Bildbreite |
| IS_GET_IMAGE_SIZE_X | Rücklesen der aktuellen Bildbreite |
| IS_GET_IMAGE_SIZE_X_MIN | Kleinster Wert für die AOI-Bildbreite |
| IS_GET_IMAGE_SIZE_X_MAX | Größter Wert für die AOI-Bildbreite |
| IS_GET_IMAGE_SIZE_X_INC | Schrittweite für die AOI-Bildbreite |
| IS_GET_IMAGE_SIZE_Y | Rücklesen der aktuellen Bildhöhe |
| IS_GET_IMAGE_SIZE_Y_MIN | Kleinster Wert für die AOI-Bildhöhe |
| IS_GET_IMAGE_SIZE_Y_MAX | Größter Wert für die AOI-Bildhöhe |
| IS_GET_IMAGE_SIZE_Y_MINC | Schrittweite für die AOI-Bildhöhe |
| **y** | |
| 1…yMax | Bildhöhe |

**Rückgabewert:**

In Verbindung mit *IS_GET_IMAGE_SIZE_X* und *IS_GET_IMAGE_SIZE_Y* werden die aktuellen Einstellungen gelesen, sonst *IS_SUCCESS* oder *IS_NO_SUCCESS*.

## 4.113.is_SetIO (nur UI-1543-M)

**Syntax:**

INT is_SetIO (HIDS hf, INT nIO)

**Beschreibung:**

*is_SetIO()* setzt die beiden zusätzlichen digitalen Ausgänge, oder liest die momentanen Einstellungen zurück.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **nIO** | Bitmaske für Ausgänge |
| 0x00 (00) | Beide Ausgänge auf 0 |
| 0x01 (01) | Erster Ausgang auf 1 zweiter auf 0 |
| 0x02 (10) | Erster Ausgang auf 0 zweiter auf 1 |
| 0x03 (11) | Beide Ausgänge auf 1 |
| IS_GET_IO | Rückgabe der momentanen Bitmaske |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*, aktueller Zustand in Verbindung mit *IS_GET_IO*.

## 4.114. is_SetKeyColor

**Syntax:**

INT is_SetKeyColor (HIDS hf, INT r, INT g, INT b)

**Beschreibung:**

Mit *is_SetKeyColor()* wird die Key-Farbe für den DirectDraw Overlay-Surface-Modus definiert. Die Funktion dient ebenfalls dazu, die Key-Farbe auszulesen. Der auszulesende Farbwert wird über den Parameter r übergeben. Als Ergebnis liefert die Funktion, je nach Aufruf, entweder den Wert eines Farbanteils (0...255) oder den RGB-Wert (0 ... 16777215).

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **r** | Rot-Anteil der Keying-Farbe (0...255). |
| IS_GET_KC_RED | gibt den Rot-Anteil zurück |
| IS_GET_KC_GREEN | gibt den Grün-Anteil zurück |
| IS_GET_KC_BLUE | gibt den Blau-Anteil zurück |
| IS_GET_KC_RGB | gibt das RGB-Tripple zurück |
| **g** | Grün-Anteil der Keying-Farbe (0...255). |
| **b** | Blau-Anteil der Keying-Farbe (0...255). |

**Rückgabewert:**

Farbwert in Verbindung mit *IS_GET_KC_RGB, IS_GET_KC_RED, IS_GET_KC_GREEN, IS_GET_KC_BLUE* sonst *IS_SUCCESS, IS_NO_SUCCESS*

 

## 4.115. is_SetLED

**Syntax:**

INT is_SetLED (HIDS hf, INT nValue)

**Beschreibung:**

Mit *is_SetKeyColor()* wird die LED auf der Gehäuserückseite der Kamera ein-/ausgeschaltet.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **nValue** | |
| IS_SET_LED_OFF | Schaltet LED aus. |
| IS_SET_LED_ON | Schaltet LED an. |
| IS_SET_LED_TOGGLE | Zwischen *AN* und *AUS* umschalten. |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.116. is_SetMemoryMode

**Syntax:**

INT is_SetMemoryMode (HIDS hf, INT nCount, INT nDelay)

**Beschreibung:**

Mit der Funktion *is_SetMemoryMode()* wird die Bildaufnahme über das optionale Memoryboard aktiviert. Als Parameter wird die Anzahl der Bilder, die in den Speicher aufgenommen werden sollen übergeben. Bei Verwendung des Pre-Trigger Modus gibt dieser Parameter die Größe des Ringspeichers an. Wird für *nCount* die Konstante *IS_MEMORY_DISABLE* übergeben, wird das Memoryboard deaktiviert.
Mit dem Parameter *nDelay* wird der Abstand in Millisekunden in dem Bilder in den Speicher übertragen werden sollen angegeben. Wird für den Parameter *nDelay* die Konstante *IS_MEMORY_USE_TRIGGER* übergeben, wird im Sequenz-Mode zwischen zwei Aufnahmen auf einen neuen Trigger gewartet. Im Pre-Trigger Modus wird dieser Parameter ignoriert (siehe auch *3.12 Memory Handling der Kamera*).

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **nCount** | Gibt die Anzahl an Bildern an, die im Post-Triggermodus in den Speicher übertragen werden.<br>Im Pre-Triggermodus, wird hiermit die Größe des Ringspeichers festgelegt. |
| IS_MEMORY_MODE_DISABLE | Deaktiviert das Memoryboard |
| IS_MEMORY_GET_COUNT | Gibt die eingestellte Bildanzahl zurück. |
| IS_MEMORY_GET_DELAY | Gibt die Verzögerung zwischen zwei Bildern im Post-Triggermodus zurück. |
| **nDelay** | Stellt die Verzögerung in Millisekunden zwischen zwei aufgenommenen Bildern im Post-Triggermodus ein. Die Bilderfassung erfolgt über einen intern gesteuerten Software Trigger. |
| IS_MEMORY_USE_TRIGGER | Wartet bei einer Sequenzaufnahme auf den nächsten Trigger. |

**Rückgabewert:**

*IS_SUCCESS,*
*IS_NO_MEMORY_BOARD_CONNECTED*
*IS_TOO_LESS_MEMORY*
aktuelle Einstellung in Verbindung mit *IS_MEMORY_GET_COUNT* oder *IS_MEMORY_GET_DELAY*

Nachdem der Memorymodus aktiviert wurde, können Kameraparameter, die die Bildgröße beeinflussen nicht mehr geändert werden. Dazu gehören folgende Funktionen:

- is_SetWindowPos
- is_SetWindowSize

## 4.117. is_SetPixelClock

**Syntax:**

INT is_SetPixelClock (HIDS hf, INT Clock)

**Beschreibung:**

*is_SetPixelClock()* setzt die Frequenz, mit der die Bilddaten aus dem Sensor ausgelesen werden. Eine zu hohe Frequenz kann dazu führen, dass Bilder während der Übertragung verloren gehen. Durch Ändern des Pixelclocks, ändert sich auch die Framerate und die Exposure-Zeit. Der aktuelle Bildeinzug wird dabei abgebrochen.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **Clock** | Pixelclock in MHz |
| IS_GET_PIXEL_CLOCK | Aktueller Pixelclock |
| IS_GET_DEFAULT_PCLK | Gibt den Standard-Pixeltakt zurück |

**Rückgabewert:**

Aktuelle Einstellung in Verbindung mit *IS_GET_PIXEL_CLOCK*, sonst *IS_SUCCESS* oder *IS_NO_SUCCESS*

## 4.118. is_SetRopEffect

**Syntax:**

INT is_SetRopEffect (HIDS hf, INT effect, INT param, INT reserved)

**Beschreibung:**

*is_SetRopEffect()* stellt den zu verwendenden Bildmanipulationseffekt ein.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **effect** | |
| IS_SET_ROP_MIRROR_UPDOWN | Spiegelt das komplette Bild in Echtzeit um die horizontale Achse. |
| IS_SET_ROP_MIRROR_LEFTRIGHT | Spiegelt das Bild in der Kamera an der vertikalen Achse.<br>Diese Funktion ist, je nach Kamera, eine Hardware- oder eine Software-Funktion. |
| IS_GET_ROP_EFFECT | Gibt die aktuelle Einstellungen zurück |
| **param** | Ein/Ausschalten des Rop-Effekts<br>0 = Ausschalten<br>1 = Einschalten |
| **reserved** | Nicht verwendet. |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS* oder die aktuellen Einstellungen bei *IS_SET_ROP_EFFECT*

## 4.119. is_SetSaturation

**Syntax:**

INT is_SetSaturation (HIDS hf, INT ChromU, INT ChromV)

**Beschreibung:**

Über *is_SetSaturation()* kann die Software-Farbsättigung eingestellt werden. Diese Funktion arbeitet nur mit dem Farbformat YUV.

**Übergabeparameter:**

| | |
|---|---|
| **Hf** | Handle auf Kamera |
| **ChromU** | Sättigung-U: Wert multipliziert mit 100.<br>Bereich: [*IS_MIN_SATURATION* … *IS_MAX_SATURATION*] |
| IS_GET_SATURATION_U | Gibt den aktuellen Wert für die U-Sättigung zurück |
| **ChromV** | Sättigung-V: Wert multipliziert mit 100.<br>Bereich: [*IS_MIN_SATURATION* … *IS_MAX_SATURATION*] |
| IS_GET_SATURATION_V | Gibt den aktuellen Wert für die V-Sättigung zurück |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*
*IS_INVALID_PARAMETER* (ungültiger Wert für den Parameter *ChromU* oder *ChromV*)
Aktuelle Einstellung in Verbindung mit *IS_GET_SATURATION_U* oder *IS_GET_SATURATION_V*.

## 4.120. is_SetSubSampling

**Syntax:**

INT is_SetSubSampling (HIDS hf, INT mode)

**Beschreibung:**

Mit *is_SetSubSampling()* kann sowohl in horizontaler als auch in vertikaler Richtung der Subsampling-Modus aktiviert werden. Je nach Sensor ist neben zweifachem auch vierfaches Subsampling möglich.
Bei zweifachem Subsampling wird jeder zweite Pixel übersprungen, wodurch sich die Bildgröße je Subsamplingrichtung halbiert und die Framerate erhöht. Dies entspricht einer Hardware Skalierung des Bildes von 2:1. Bei Farbkameras wird aus technischen Gründen 4:2 Subsampling verwendet, bei dem jeweils zwei Pixel ausgelesen und zwei Pixel übersprungen werden.
Der Sensor der UI-154x-M führt bauartbedingt nur ein Color-Subsampling durch. Dies kann bei feinen Bildstrukturen zu leichten Artefakten führen.

Monochrom-Sensor

Abb. 13: zweifaches Subsampling

Color-Sensor

Abb. 14: zweifaches Subsampling

Abb. 15: vierfaches Subsampling

Abb. 16: vierfaches Subsampling

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **mode** | |
| IS_SUBSAMPLING_DISABLE | Deaktiviert das Subsampling. |
| IS_SUBSAMPLING_2X_VERTICAL | Aktiviert zweifaches Subsampling in vertikaler Richtung. |
| IS_SUBSAMPLING_4X_VERTICAL | Aktiviert vierfaches Subsampling in vertikaler Richtung. |
| IS_SUBSAMPLING_2X_HORIZONTAL | Aktiviert zweifaches Subsampling in horizontaler Richtung. |
| IS_SUBSAMPLING_4X_HORIZONTAL | Aktiviert vierfaches Subsampling in horizontaler Richtung. |
| IS_GET_SUBSAMPLING | Gibt die aktuelle Einstellung zurück. |
| IS_GET_SUBSAMPLING_TYPE | Gibt zurück, ob die Kamera farberhaltendes Subsampling (*IS_SUBSAMPLING_COLOR*) verwendet oder nicht. |
| IS_GET_SUPPORTED_SUBSAMPLING | Gibt die unterstützten Subsampling Modi zurück. |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*, oder aktuelle Einstellung bei *IS_GET_ SUBSAMPLING*

## 4.121. is_SetTestImage

**Syntax:**

INT is_SetTestImage (HIDS hf, INT nMode)

**Beschreibung:**

Die Funktion *is_SetTestImage()* schaltet verschiedene vordefinierte Testbilder im Memoryboardbetrieb ein. Wurde ein bestimmter Modus aktiviert, wird bei der nächsten Memoryaufnahme dieses Testbild anstelle der echten Bilddaten übertragen. Die Testbilder verschieben sich bei jeder Aufnahme um eine Zeile (Grauwert usw.).

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **nMode** | |
| IS_SET_TEST_IMAGE_DISABLED | Testbilder werden deaktiviert |
| IS_SET_TEST_IMAGE_MEMORY_1 | 1. Testbild wird aktiviert (einfarbige graue Rampe) |
| IS_SET_TEST_IMAGE_MEMORY_2 | 2. Testbild wird aktiviert (Bayer RGB Rampe) |
| IS_SET_TEST_IMAGE_MEMORY_3 | 3. Testbild wird aktiviert (eine weiße horizontale Linie über einem schwarzen Bild) |
| IS_GET_TEST_IMAGE | Aktuelle Einstellung wird zurückgegeben. |

**Rückgabewert:**

In Verbindung mit *IS_GET_TEST_IMAGE* wird die aktuellen Einstellungen gelesen, sonst *IS_SUCCESS* oder *IS_NO_SUCCESS*.

## 4.122. is_SetTriggerDelay

**Syntax:**

INT is_SetTriggerDelay (HIDS hf, INT nDelay)

**Beschreibung:**

Mit *is_SetTriggerDelay()* lässt sich die Verzögerungszeit zwischen Eingang eines externen Triggersignals und dem Start der Belichtung einstellen.
Die eingestellte Delayzeit wirkt sich additiv zur Defaultdelayzeit aus. Dies ist diejenige Zeit, die immer vorhanden ist, bis das externe Triggersignal bis zum Sensor durchgeroutet ist. Es gelten folgende Werte für die Defaultdelayzeit (bei TTL Signalpegel und 50% Triggerlevel):

- CMOS
  - HI_LO: 38,0µs
  - LO_HI: 19,7µs
- CCD
  - HI_LO: 61,5µs
  - LO_HI: 43,2µs

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **nDelay** | Zeit um die die Bildaufnahme verzögert wird (in µs) |
| IS_GET_TRIGGER_DELAY | Rückgabe der momentan eingestellten Verzögerung. |
| IS_GET_MIN_TRIGGER_DELAY | Gibt den minimal einstellbaren Wert zurück. |
| IS_GET_MAX_TRIGGER_DELAY | Gibt den maximal einstellbaren Wert zurück. |
| IS_GET_TRIGGER_DELAY_GRANULARITY | Gibt die Auflösung der einstellbaren Verzögerungszeit zurück. |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*, aktuelle Einstellungen in Verbindung mit *IS_GET_TRIGGER_DELAY*

## 4.123. is_SetWhiteBalance

**Syntax:**

INT is_SetWhiteBalance (HIDS hf, INT nMode)

**Beschreibung:**

*is_SetWhiteBalance()* aktiviert den mit *nMode* angegebenen Weißabgleich.

> Der Software Weißabgleich lässt sich nicht aktivieren, wenn der automatische Weißabgleich mit *is_SetAutoParameter()* eingeschaltet wurde.
>
> *is_SetWhiteBalance()* führt eine Farbskalierung per Software durch. Optimale Ergebnisse sind mit der Funktion *is_SetAutoParameter()* zu erzielen, da diese die hardwareseitigen Gainregler verwendet.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **nMode** | |
| IS_SET_WB_DISABLE | Weißabgleich deaktivieren. |
| IS_SET_WB_USER | Benutzerdefinierte Werte verwenden. (Faktoren müssen anschließend mit *is_SetWhiteBalanceMultipliers()* gesetzt werden. |
| IS_SET_WB_AUTO_ENABLE | Aktiviert automatischen Weißabgleich bei jedem neuen Bild. |
| IS_SET_WB_AUTO_ENABLE_ONCE | Automatischer Weißabgleich mit dem nächsten aufgenommenen Bild. |
| IS_SET_WB_DAYLIGHT_65 | Industriestandard Daylight 65 |
| IS_SET_WB_COOL_WHITE | Industriestandard CWF (Cool White Fluorescent) |
| IS_SET_WB_ILLUMINANT_A | Industriestandard Illuminant A |
| IS_SET_WB_U30 | Industriestandard Ultralume 30 |
| IS_SET_WB_HORIZON | Industriestandard Horizon |
| IS_GET_WB_MODE | Rückgabe des aktuellen Modus |

**Rückgabewert:**

Aktuelle Einstellung in Verbindung mit *IS_GET_WB_MODE*, sonst *IS_SUCCESS, IS_NO_SUCCESS*

 

## 4.124. is_SetWhiteBalanceMultipliers

**Syntax:**

INT is_SetWhiteBalanceMultipliers (HIDS hf, double dblRed, double dblGreen, double dblBlue)

**Beschreibung*:***

Mit *is_SetWhiteBalanceMultipliers()* werden die benutzerdefinierten Faktoren gesetzt, die für den Weißabgleich herangezogen werden. Details hierzu siehe auch bei *4.121 is_SetWhiteBalance*.

> *is_SetWhiteBalanceMultipliers()* setzt die Parameter für die Software-Farbskalierung, die mit der Funktion *is_SetWhiteBalance()* durchgeführt wird. Eine bessere Steuerung der Grundfarben kann mit der Funktion *is_SetAutoParameter()* erreicht werden.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **dblRed** | Neuer Faktor für rot |
| **dblGreen** | Neuer Faktor für grün |
| **dblBlue** | Neuer Faktor für blau |

**Rückgabewert:**

*IS_SUCCESS oder IS_NO_SUCCESS*

## 4.125. is_ShowDDOverlay

**Syntax:**

INT is_ShowDDOverlay (HIDS hf)

**Beschreibung*:***

*is_ShowDDOverlay()* blendet das Overlay im DirectDraw BackBuffer-Modus ein. Es werden die zuletzt im Overlaybuffer enthaltenen Daten dargestellt. Die Darstellung erfolgt nun über drei Bildbuffer. Je nach VGA-Karte kann jetzt die Bildwiederholrate kleiner sein als ohne Overlay-darstellung.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |

**Rückgabewert:**

*IS_SUCCESS oder IS_NO_SUCCESS*

## 4.126. is_StealVideo

**Syntax:**

INT is_StealVideo (HIDS hf, int Wait)

**Beschreibung:**

Die Funktion *is_StealVideo()* leitet das *Stehlen* eines Bildes im DirectDraw Livemodus ein. Das gestohlene Bild wird dabei in den aktuellen aktiven Bildspeicher im RAM geschrieben. Dabei wird das mit der Funktion *is_SetColorMode()* eingestellte Farbformat verwendet.
Mit der Funktion *is_PrepareStealVideo()* kann voreingestellt werden, ob das Bild umgeleitet werden muss oder kopiert werden kann. Bei Verwendung der Option *Kopieren* wird das Bild mit DirectDraw angezeigt und in den aktuell aktiven Bildspeicher kopiert.
Siehe auch *Abb. 6: Events im Livemodus* unter *3.9 Event Handling (Interrupt gesteuerter Bild-einzug)*.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **Wait** | |
| IS_WAIT | Funktion wartet bis Bild im Speicher ist. |
| IS_DON'T_WAIT | Funktion kehrt sofort zurück |

**Rückgabewert:**

IS_SUCCESS, IS_NO_SUCCESS

## 4.127. is_StopLiveVideo

**Syntax:**

INT is_StopLiveVideo (HIDS hf, INT Wait)

**Beschreibung:**

Die Funktion *is_StopLiveVideo()* stoppt den Live Modus und friert das Bild im Speicher der VGA-Karte oder im Systemspeicher des PCs ein.
Über den Übergabeparameter *Wait* wird das Rückkehrverhalten der Funktion gesteuert. Es kann zwischen den beiden Modi *Funktion kehrt sofort zurück* und *die Funktion wartet bis das letzte Bild fertig aufgenommen ist* gewählt werden. Im ersten Fall wird das Bild im Hintergrund fertig digitalisiert.
Durch die Verwendung von *IS_FORCE_VIDEO_STOP* kann eine mit *is_FreezeVideo(hf, IS_DONT_WAIT)* gestartete Einzelbildaufnahme sofort beendet werden.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **Wait** | |
| IS_WAIT | Funktion wartet bis Bild ganz im Speicher ist |
| IS_DONT_WAIT | Funktion kommt sofort zurück. |
| IS_FORCE_VIDEO_STOP | Halte Digitalisierung sofort an |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.128. is_TransferImage

**Syntax:**

INT is_TransferImage (HIDS hf, INT nMemID, INT seqID, INT imageNr, INT reserved)

**Beschreibung:**

Die Funktion *is_TransferImage()* wird verwendet um Bilder aus dem Kameraspeicher auf dem optionalen Memoryboard einzulesen. Als Parameter wird die ID des Speichers angegeben, in den das Bild übertragen wird. Außerdem wird die gewünschte Sequenz ID und die Nummer des Bildes benötigt. Werden für die Sequenz und die Bildnummer „0“ angegeben, wird das zuletzt aufgenommene Bild übertragen.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **nMemID** | ID des Speichers, muss zuvor mit *is_AllocImageMem()* oder *is_SetAllocatedImageMem()* angelegt worden sein. |
| **seqID** | Die ID der Sequenz aus der ein Bild eingelesen wird. |
| **imageNr** | Gibt die Nummer des Bildes innerhalb der Sequenz an. |

**Rückgabewert:**

*IS_SUCCESS, IS_IMAGE_NOT_PRESENT, IS_NO_SUCCESS*

## 4.129. is_TransferMemorySequence

**Syntax:**

INT is_TransferMemorySequence (HIDS hf, INT seqID, INT StartNr, INT nCount, INT nSeqPos)

**Beschreibung:**

Die Funktion *is_TransferMemorySequence()* wird verwendet um mehrere Bilder aus dem Kameraspeicher in eine SDK Sequenz einzulesen. Der SDK Sequenz muss zuvor mit *is_AddToSequence()* genügend mit *is_AllocateMemory()* angelegter Speicher hinzugefügt worden sein. Als Parameter wird die ID der Kameraspeicher Sequenz benötigt. Außerdem kann die Startnummer ab der Bilder eingelesen werden angegeben werden. Mit *nCount* wird festgelegt wie viele Bilder ab *StartNr* übertragen werden sollen.
Der letzte Parameter dient dazu die Bilder ab *nSeqPos* in die Sequenz zu übertragen.

**Beispiel:**

Übertragen von 3 Bildern aus der Kamerasequenz „1" ab Bild Nummer „2" in die SDK Sequenz ab Position „3".
Aufruf: `is_TransferMemorySequence(hf, 1, 2, 3, 3);`

| SDK Sequenz |
| --- |
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |

| Kameraspeicher (SeqNr 1) |
| --- |
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |

**Übergabeparameter:**

| | |
| --- | --- |
| **hf** | Handle auf Kamera |
| **seqID** | Die ID der Sequenz im Kameraspeicher, aus der die Bilder eingelesen werden. |
| **StartNr** | Bildindex ab dem Bilder aus der Sequenz eingelesen werden. |
| **nCount** | Anzahl an Bildern, die aus dem Speicher in die Sequenz übertragen werden.<br>(0 = Alle Bilder der Sequenz ab *StartNr* werden übertragen). |
| **nSeqPos** | Anfangsposition der Sequenz |

**Rückgabewert:**

*IS_SUCCESS, IS_IMAGE_NOT_PRESENT, IS_NO_SUCCESS*

## 4.130. is_UnlockDDMem

**Syntax:**

INT is_UnlockDDMem (HIDS hf)

**Beschreibung:**

*is_UnlockDDMem()* hebt den Zugriff auf den Bildspeicher in den DirectDraw Modi auf. Anschließend wird der Inhalt des Backbuffers auf dem Bildschirm aktualisiert.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.131. is_UnlockDDOverlayMem

**Syntax:**

INT is_UnlockDDOverlayMem (HIDS hf)

**Beschreibung:**

*is_UnlockDDOverlayMem()* hebt den Zugriff auf den Overlay-Buffer im DirectDraw BackBuffer Modus wieder auf. Anschließend wird der Inhalt des Overlay-Buffers auf dem Bildschirm aktualisiert (wenn mit *is_ShowDDOverlay()* das Overlay eingeblendet ist).

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.132. is_UnlockSeqBuf

**Syntax:**

INT is_UnlockSeqBuf (HIDS hf, INT nNum, char* pcMem)

**Beschreibung:**

Mit *is_UnlockSeqBuf()* wird die Bildaufnahme in einen zuvor gelockten Bildspeicher wieder erlaubt. Der Bildspeicher wird wieder an der früheren Position der Sequenzliste eingefügt.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **nNum** | Nummer des Bildspeichers, der freigegeben werden soll (1...max) |
| **pcMem** | Startadresse des Bildspeichers |

nNum bezeichnet die Position in der Sequenzliste und nicht die mit *is_AllocImageMem()* vergebene Speicher-ID.

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.133. is_UpdateDisplay

**Syntax:**

INT is_UpdateDisplay (HIDS hf)

**Beschreibung:**

*is_UpdateDisplay()* aktualisiert manuell die Bildschirmausgabe in den DirectDraw Modi. Die Aktualisierung erfolgt normalerweise automatisch durch den Treiber. Es kann in wenigen Fällen notwendig werden die Bildschirmausgabe manuell auf geänderte Gegebenheiten anzupassen. (Siehe auch Funktion *4.93 is_SetDisplayMode*)

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.134. is_WriteEEPROM

**Syntax:**

INT is_WriteEEPROM (HIDS hf, INT Adr, char* pcString, INT Count)

**Beschreibung:**

Auf der uEye befindet sich ein EEPROM als kleiner Speicher. Neben den fest auf der Karte hinterlegten Informationen können 64 Byte eigene Daten in das EEPROM geschrieben werden. Mit der Funktion *is_ReadEEPROM()* kann der Inhalt dieses 64-Byte Blockes gelesen werden.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **Adr** | Anfangsadresse, ab der Daten geschrieben werden sollen (0...63) |
| **pcString** | Zeichenkette, die eigene Daten enthält |
| **Count** | Anzahl zu schreibender Zeichen (1...64) |

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS*

## 4.135. is_WriteI2C (nur uEyeLE)

**Syntax:**

INT is_WriteI2C (HIDS hf, INT nDeviceAddr, INT nRegisterAddr, BYTE* pbData, INT nLen)

**Beschreibung:**

Mit *is_WriteI2C()* können Daten über den $I^2C$-Bus geschrieben werden. Der $I^2C$-Bustakt beträgt 100 kHz.

**Übergabeparameter:**

| | |
|---|---|
| **hf** | Handle auf Kamera |
| **nDeviceAdr** | Slave address |
| **nRegisterAddr** | Register address (nur 8 Bit-Adressen gültig). |
| **data** | Zu schreibende Daten |
| **length** | Datenlänge |

> Ungültige Adresssen für *nRegisterAddr*:
> 0x48, 0x4C, 0x51, 0x52, 0x55, 0x5C, 0x5D, 0x69 (diese werden intern verwendet).

**Rückgabewert:**

*IS_SUCCESS, IS_NO_SUCCESS* or *IS_INVALID_I2C_DEVICE_ADDRESS*

# 5. Fehlermeldungen

| Nr | Fehler | Beschreibung |
| --- | --- | --- |
| -1 | IS_NO_SUCCESS | allgemeine Fehlermeldung |
| 0 | IS_SUCCESS | allgemeine Erfolgsmeldung - kein Fehler |
| 1 | IS_INVALID_CAMERA_HANDLE | Das Kamera Handle ist ungültig. Für die meisten der Funktionen des uEye SDk wird das Kamera Handle als erster Parameter erwartet. |
| 2 | IS_IO_REQUEST_FAILED | Eine IO Anforderung des uEye Treibers schlug fehl. Eventuell passen Api dll und Treiber Datei nicht zusammen. |
| 3 | IS_CANT_OPEN_DEVICE | Ein Versuch die Kamera zu öffnen schlug fehl (Kamera nicht vorhanden oder Fehler beim Initialisieren). |
| 11 | IS_CANT_OPEN_REGISTRY | Fehler beim Öffnen eines Windows Registry Keys |
| 12 | IS_CANT_READ_REGISTRY | Fehler beim Lesen von Einstellungen aus der Windows Registry |
| 15 | IS_NO_IMAGE_MEM_ALLOCATED | Der Treiber konnte keinen Speicher reservieren. |
| 16 | IS_CANT_CLEANUP_MEMORY | Der Treiber konnte den verwendeten Speicher nicht freigeben. |
| 17 | IS_CANT_COMMUNICATE_WITH_DRIVER | Kommunikation mit dem Treiber schlug fehl, weil kein Treiber geladen ist. |
| 49 | IS_INVALID_MEMORY_POINTER | Ungültiger Zeiger oder ungültige SpeicherID |
| 50 | IS_FILE_WRITE_OPEN_ERROR | Datei kann nicht zum Schreiben geöffnet werden. |
| 51 | IS_FILE_READ_OPEN_ERROR | Datei kann nicht zum Lesen geöffnet werden. |
| 52 | IS_FILE_READ_INVALID_BMP_ID | Die angegebene Datei ist kein gültiges Bitmap |
| 53 | IS_FILE_READ_INVALID_BMP_SIZE | Die Größe des Bitmaps ist falsch (zu groß). |
| 108 | IS_NO_ACTIVE_IMG_MEM | Kein aktivierter Bildspeicher vorhanden. Der Speicher muss mit der Funktion is_SetActiveImageMem aktiviert werden, oder es muss mit der Funktion is_AddTo Sequence eine Sequenz aufgebaut werden. |
| 112 | IS_SEQUENCE_LIST_EMPTY | Die Sequenzliste ist leer und kann nicht gelöscht werden. |
| 113 | IS_CANT_ADD_TO_SEQUENCE | Der Bildspeicher befindet sich bereits in der Sequenz und kann nicht doppelt hinzugefügt werden. |
| 117 | IS_SEQUENCE_BUF_ALREADY_LOCKED | Der Speicher konnte nicht gelockt werden. Der Zeiger auf den Buffer ist ungültig. |
| 118 | IS_INVALID_DEVICE_ID | Die Device ID ist ungültig. Gültige IDs liegen zwischen 0 und 255. |
| 119 | IS_INVALID_BOARD_ID | Die Board ID ist ungültig. Gültige IDs liegen zwischen 1 und 255. |
| 120 | IS_ALL_DEVICES_BUSY | Alle Kameras sind in Verwendung |
| 122 | IS_TIMED_OUT | Ein Timeout trat auf. Eine Bildaufnahme konnte nicht in der vorgeschriebenen Zeit beendet werden. |
| 125 | IS_INVALID_PARAMETER | Einer der übergebenen Parameter ist ausserhalb des gültigen Bereichs, oder für diesen Sensor nicht unterstützt, bzw in diesem Modus nicht zugänglich. |
| 127 | IS_OUT_OF_MEMORY | Es konnte kein Speicher allokiert werden. |
| 139 | IS_NO_USB20 | Die Kamera ist an einem Port verbunden, der nicht den High Speed Standard USB 2.0 unterstützt. Kameras ohne Speicher können nicht an USB 1.1 Ports betrieben werden. |
| 140 | IS_CAPTURE_RUNNING | Es läuft bereits eine Aufnahme, die beendet werden muss, bevor eine neue begonnen werden kann. |

| Nr | Fehler | Beschreibung |
|---|---|---|
| 141 | IS_MEMORY_BOARD_ACTIVATED | Der Memorymodus ist aktiv. Deshalb können keine Änderungen and der Bildgröße vorgenommen werden. |
| 143 | IS_NO_MEMORY_BOARD_CONNECTED | Es wurde kein Memoryboard detektiert. Die Funktion erfordert eine Kamera mit Memoryboard. |
| 144 | IS_TOO_LESS_MEMORY | Die eingestellte Anzahl Bilder überschreitet in der aktuellen Größe die Kapazität des Memoryboards (4MB) |
| 145 | IS_IMAGE_NOT_PRESENT | Das angeforderte Bild ist nicht im Kameraspeicher vorhanden oder nicht mehr gültig. |
| 147 | IS_MEMORYBOARD_DISABLED | Das Memoryboard wurde aufgrund eines Fehlers deaktiviert. Die Funktion steht nicht mehr zur Verfügung. |
| 148 | IS_TRIGGER_ACTIVATED | Die Funktion ist nicht möglich, da die kamera auf ein Triggersignal wartet. |
| 151 | IS_CRC_ERROR | Beim Lesen der Einstellungen trat ein CRC Fehler auf. |
| 152 | IS_NOT_YET_RELEASED | Diese Funktion ist in dieser Treiberversion noch nicht freigeschaltet. |
| 153 | IS_NOT_CALIBRATED | Die Kamera enthält keine Kalibrierungsdaten. |
| 154 | IS_WAITING_FOR_KERNEL | Es wird noch auf eine Rückmeldung des Kerneltreibers gewartet. |
| 155 | IS_NOT_SUPPORTED | Das verwendete Kameramodell unterstützt diese Funktion oder Einstellung nicht. |
| 157 | IS_OPERATION_ABORTED | Es konnte keine Datei gespeichert werden, weil der Dialog ohne Auswahl beendet wurde. |
| 158 | IS_BAD_STRUCTURE_SIZE | Eine interne Struktur hat die falsche Größe. |
| 159 | IS_INVALID_BUFFER_SIZE | Der Bildspeicher hat die falsche Größe um das Bild im gewünschten Format aufzunehmen. |
| 160 | IS_INVALID_PIXEL_CLOCK | Diese Einstellung ist bei dem aktuell eingestellten Pixelclock nicht möglich. |
| 161 | IS_INVALID_EXPOSURE_TIME | Diese Einstellung ist bei der aktuell eingestellten Belichtungszeit nicht möglich. |
| 162 | IS_AUTO_EXPOSURE_RUNNING | Die Einstellung kann nicht geändert werden, solange die automatische Belichtungszeitregelung aktiviert ist. |
| 163 | IS_CANNOT_CREATE_BB_SURF | BackBuffer Surface kann nicht angelegt werden. |
| 164 | IS_CANNOT_CREATE_BB_MIX | BackBuffer Mixer Surface kann nicht angelegt werden. |
| 165 | IS_BB_OVLMEM_NULL | BackBuffer Overlay Speicher kann nicht gelocked werden. |
| 166 | IS_CANNOT_CREATE_BB_OVL | BackBuffer Overlay Speicher kann nicht erzeugt werden. |
| 167 | IS_NOT_SUPP_IN_OVL_SURF_MODE | Wird im BackBuffer Overlay Modus nicht unterstützt. |
| 168 | IS_INVALID_SURFACE | BackBuffer Surface ungültig. |
| 169 | IS_SURFACE_LOST | BackBuffer Surface lost. |
| 170 | IS_RELEASE_BB_OVL_DC | Fehler bei der Freigabe des Overlay Device Context. |
| 171 | IS_BB_TIMER_NOT_CREATED | BackBuffer Timer konnte nicht angelegt werden. |
| 172 | IS_BB_OVL_NOT_EN | BackBuffer Overlay wurde nicht aktiviert. |
| 173 | IS_ONLY_IN_BB_MODE | Nur im BackBuffer Modus möglich. |
| 174 | IS_INVALID_COLOR_FORMAT | Ungültiges Farbformat. |

# Service und Support

Um Ihre Anfrage effektiv und schnell bearbeiten zu können, benötigen wir von Ihnen die folgenden Daten per Fax (07134/96196-99):

Anschrift: ____________________

Name: ____________________

Tel.: ____________________ Fax: ____________________

E-Mail.: ____________________

Kameramodell: ____________________

O mit Gehäuse (0) O ohne Gehäuse (1) O Platinenversion (2)

O Monochrom (M) O Farbe (C)

O OnBoardMemory (M)

O Andere: ____________________

Seriennummer: ____________________ Treiberversion: ____________________

Kaufdatum: __________ Gekauft bei: ____________________

Beabsichtigter Einsatz: ____________________

Läuft das Demoprogramm *uEye Demo?* O ja O nein

Erstinstallation? O ja O nein

Kamera plötzlich ausgefallen? O ja O nein

Erklärung: ____________________

Wurde die Kamera an einem anderen Rechner gegengeprüft? O ja O nein

Beschreibung des Problems:

____________________

____________________

____________________

____________________

Rechnertyp: ____________________ Motherboard: ____________________

CPU: ____________________ RAM: ____________________

VGA-Karte Typ: ____________________ VGA-Speicher: ____________________

SCSI-Controller? O ja O nein

Netzwerk: O ja O nein Typ: ____________________

USB onboard? O ja O nein

USB Einsteckkarte: O ja O nein Typ: ____________________

Betriebssystem: O Win2000 installiertes Service-Pack: __________

O WinXP installiertes Service-Pack: __________

O Linux Distribution: __________ Kernelversions-Nr.: __________

# Abbildungsverzeichnis

Abb. 1:Prinzipieller Aufbau des Bayer-Pattern ....................................................3
Abb. 2: Bitmap Modus ....................................................4
Abb. 3: DirectDraw BackBuffer Modus ....................................................5
Abb. 4: DirectDraw Overlay-Surface-Modus ....................................................5
Abb. 5: Events bei Einzeltriggeraufnahme ....................................................11
Abb. 6: Events im Livemodus ....................................................11
Abb. 7: Events im Memorymodus....................................................12
Abb. 8: Pre-Trigger Modus ....................................................15
Abb. 9: Post-Trigger Modus....................................................17
Abb. 10: Anhängender Speichermodus....................................................18
Abb. 11: Ablauf Direct mode / Memory mode ....................................................19
Abb. 12: Timing Diagramm Memoryboard....................................................19
Abb. 13: zweifaches Subsampling....................................................135
Abb. 14: zweifaches Subsampling....................................................135
Abb. 15: vierfaches Subsampling ....................................................135
Abb. 16: vierfaches Subsampling ....................................................135

# Tabellenverzeichnis

Tabelle 1: CAMINFO Datenstruktur des EEPROMS....................................................2
Tabelle 2: Farb- und Speicherformate....................................................3
Tabelle 3: Funktionsliste Initialisierung und Terminierung ....................................................7
Tabelle 4: Funktionsliste Bilderfassung und Speichermanagement ....................................................7
Tabelle 5: Funktionsliste Auswahl der Betriebsmodi und Rücklesen der Einstellungen....................................................9
Tabelle 6: Funktionsliste Double- und Mehrfach-Buffering ....................................................9
Tabelle 7: Funktionsliste Lesen und Schreiben des EEPROMS....................................................9
Tabelle 8: Funktionsliste Speichern und Laden von Bildern ....................................................9
Tabelle 9: Funktionsliste Bildausgabe....................................................9
Tabelle 10: Funktionsliste Zusätzliche DirectDraw-Funktionen ....................................................10
Tabelle 11: Funktionsliste Event Handling ....................................................10
Tabelle 12: Funktionsliste Steuerung der Ein- / Ausgänge ....................................................12
Tabelle 13: Funktionsliste $I^2C$-Funktionen....................................................12
Tabelle 14: Funktionsliste Memory Handling ....................................................13
Tabelle 15: Gültigkeit der Funktionen....................................................21